PB001: Uvod do IT

* nejpermanentnejsi ulozeni dat je hieroglyfy do kamene - u soucasnych technologii nevime jak dlouho vydrzi (existuji pouze kratce), navic diky vyvoji uz casto neexistuje odpovidajici zarizeni na prehrani nebo rozlusteni formatu souboru
  + jak to zaridit? jedna velka otazka IT
* teprve od konce osmdesatych let se pocitac stava “osobni” - do te doby pouze instituce
* opticke site klidne 6,5 TB/s (bezi na ruznych barvach)
* ukazuje se, ze nejvetsim vyuzitim virtualni reality je v zabavnim prumyslu
* standardni pohled na PC: kastla s vnitrnim vybavenim
  + dalsi pohledy: bunecne automaty, neuronove pocitace, DNA pocitace
* duvod vzniku distribuovanych systemu: narazeni na strop zvyseni vykonu (dnes: nevime jak efektivne uchladit) → misto jednoho procesoru jich mame tisic
  + otazky: jak udelat co nejrychlejsi komunikaci? (napr. prvni Cray byl sestiuhelnikovy)
* gridy - jsou velmi rozsahle! (napr. pres cely kontinent)
* volne propojene systemy jsou obvykle propojene specialni siti (napr. [infiniband](http://en.wikipedia.org/wiki/InfiniBand) - vyssi rychlost, 10x nizsi latence nez normalni sit)
* zpozdeni komunikace: uzce - desitky nanosekund, volne - jednotky mikrosekund, distribuovane - stovky mikrosekund (VELKY rozdil!)
* programove vybaveni nam umoznuje s pocitacem pracovat
* vrstvy OS:
  + technicke vybaveni - prace s nim
  + sprava pameti, sprava periferii, sprava souboru a dat
  + uzivatelske prostredi
  + sprava procesu!
* proces = bezici program
* programovaci jazyky
  + imperativni = prikazujici (pisou se presne instrukce)
  + deklarativni = nepredepisuji jak se ma delat, ale co se ma delat ( → vetsi prostor pro predchozi optimalizaci)
  + produkcni systemy = dobre programovani vyvozovacich retezcu - rozhodovani podle vstupnich hodnot
* aritmeticky if - if, ktery kdyz se vyhodnotil na -1, tak skocil na prvni navesti; kdyz na 0, tak na druhe navesti; kdyz 1, tak na treti navesti
* zdrojovy kod → preklad → sestaveni
* interprety - abstraktni pocitac (virtualni), vhodne pro slozite operace
  + napr. Java (JVM)
  + neni vhodny pro vyssi vypocty (pouzivame je pro prototypovani - napr. MatLab)
* Just in time prekladac - neprekladame cely program (pouze kousky), protoze predpokladam, ze nebudu program pouzivat cely; pokud je potreba vic, tak se prelozi az pri behu PC
  + resil se tak nedostatek pameti (napr. v 80. letech)
* existuje souvislost mezi architekturou a jazyky → jazyky jsou tvoreny na zaklade architektury (napr. von Neumannova architektura)
* vypocetni model
  + datove orientovany - primarni jsou data a my se starame, jak se data v prubehu programu meni (definujeme presne procesy jak se meni - napr. jako v receptu) a vysledkem je hotovy produkt
  + objektovy - nadstavba datove orientovaneho; muze koncit nejenom daty, ale i ve stavech objektu
  + funkcionalni - na zacatku mnozina funkcni a dat, na konci data
  + logicky - pracuje s tvrzenimi
* aplikace davaji pocitacum smysl (duvod proc jsou pocitace)
  + vznik PC inicializovaly vedecko-technicke vypocty (puvodne urceno pro vedu a vojenstvi)
    - napr. vyvoj atomove bomby (ackoliv prvni byla sestrojena a vypocitana lidmi - byla hromada lidi, kteri delali casti vypoctu; delalo se vse dvakrat, aby se predeslo chybam → logika teto cinnosti je v podstate logikou pocitacu a imperativnich jazyku)
  + vznik dernych stitku: Hollerith (kvuli scitani lidi v USA - automaticke trizeni) → vznik IBM
    - prvni pouziti mimo vedu → nasledne rozsireni pocitacu do jinych sfer
* vyuziti PC:
  + veda, zpracovani informaci, zabava, rizeni (management stroju i lidi, real-time systemy)

# Ciselne soustavy

* pocitaci stoji na dvojkove soustave
  + my lide pracujeme s desitkovou
* jsou mezi sebou prevoditelne
  + u celych cisel beze ztraty presnosti (u realnych neco ztracime)
* prvni pocitace - desitkova → ale rychle se opustily
* dvojkova soustava
  + dva stavy - ANO/NE (1, 0)
  + nejvetsi problemy u realnych cisel
    - rozlisitelnost - nejmensi zobrazitelne nenulove cislo
    - presnost - rozsah bitu, ktere mame k dispozici
    - zobrazuje se pomoci *mantisy* a *exponentu* (format dle IEE 754 - 60. leta)
      * stejna smernice nam sjednotila zaokrouhlovani
    - normalizace mantisy - nejvyssi bit vzdy 1 → nezapisujeme ji
      * nekdy se pouziva i nenormalizovana podoba, protoze nam umozni dosahnout vetsiho rozsahu (az k 2150)
  + zaporna cisla
    - primy kod
      * 2 nuly
      * horsi implementace aritmetickych operaci
    - inverzni kod
      * taky 2 nuly
      * delame to jako predpripravu k doplnkovem kodu
    - doplnkovy kod
      * pouze 1 nula
      * nesymetricky rozsah
      * je skutecne pouzivan v pocitacich - protoze zakladni operace *per bit* jsou jednodussi (napr. scitani je zde pouze normalni *XOR*)
  + rozahy se urcuji podle poctu bitu
* ostatni soustavy se pouzivaji predevsim pro “hutny” zapis binarnich cisel
  + vyssi nez 16 se nepouziva, kvuli snadne zapamatelnosti (protoze pamatovat si 23 dalsich znaku uz neni trivialni (skoro cela abeceda)

# Operacni systemy

* na zacatku byl v PC jen trivialni proram (Bootstrap loader), ktery pouze nahral uzivatelsky program do pocitace a udelal ho neomezenym panem nad hardwarem (uplne zacatky, kdy se spoustela pouze jedna aplikace)
* *spooling* - nezavisle zavadeni programu a jeho vykonavani
  + vyzaduje DMA (direct memory access)
* *timesharing* - virtualizace PC (resp. CPU)
  + nas pocitac je tvoren mnozinou mensich pocitacu, ktery kazdy z nich dokaze zpracovat nejakou ulohu
  + zpracovani interaktivnich uloh
  + umoznil multiuzivatelske pocitace
* ucel: zjednoduseni prace s pocitacem, sdileni zdroju
* problem casovani
  + periferie jsou vyrazne pomalejsi nez procesor → spravne prokladani IO operaci a vypoctu
    - kdyz se ceka na disk, tak procesor neco pocita
  + operacni system tudiz ma procesy, mezi kterymi samovolne prepina podle priority
* pamet
  + vetsina pameti nevyuzita (program pri svem behu travi cas vetsinou v nejakem cyklu nebo cekanim na IO/uzivatele - je “zivy” pouze na nejakem useku pameti) → kazdemu procesu dame k dispozici “celou pamet” a necinne casti jakoby nechavame na disku a vzdy nacitame jen tu “zivou” cast
  + muzeme mit vice procesu (s celkovym vetsim souctem pameti) nez je fyzicka pamet (swapovani)
  + strankovani - misto toho, aby kazda aplikace mela souvislou pamet o velikosti jakou potrebuje, tak je cela pamet “rozstrankovana” (tzn. kazdy proces dostane urcity pocet stranek jejichz soucet je velmi blizky (nebo stejny) pameti, kterou potrebuje) → pamet aplikace nemusi byt souvisla (mezery jsou stejne velke a tim padem je umime snadno zaplnit)
  + segmentace - mezikrok mezi strankovanim a swapovanim celych procesu; obsahuje logiku uvnitr programu (rika: jesltize chces odstranit tuto stranku, tak smaz i tyto, protoze bez teto ten zbytek nema smysl)
* zakladni ukolu
  + procesy a jejich sprava
  + pamet a jeji sprava
  + obsluha periferii
  + system souboru
  + ochrana a bezpecnost
* proces = bastrakce prochodu programem
  + program, ktery nam bezi na nejakem pocitaci
  + sekvencni (program = 1 proces)/paralelni (program > 1 proces) model
  + proces ma interni stav
    - programovy citac = misto, kde se v kodu nachazime
    - zasobnik = pametovy prostor, kam si ukladame co jsme volali (procedury, funkce)
    - vlastni pamet na data (halda) = jsou tam nase data
  + typy
    - klasicke (heavy-wwight)
      * vsechna data provatni
    - lehke (light-weight) - vlakno ( → Multithreading)
      * minimum vlastni pameti
      * vetsina dat sdilena
  + vytvoreni
    - operace jadra fork() - zkopirovani procesu → nahrazeni jeho dat novymi
      * presna kopie puvodniho procesu
      * rodic a potomek
    - prvni proces v OS (linux: init) vznika nestandardne...zbytek uz normalne
      * prvni vznika pozustatkem *bootstrap loaderu*
  + stavy
    - start, pripraven {ready}, bezi {running}, je blokovan - ceka {waiting}, ukoncil
  + zaroven muze bezet tolik procesu, kolik mame procesoru (jader)
  + synchronizace
    - kdyz souperi vice vlaken o jedno miste v pameti
    - kriticka sekce
      * situace, kdy rikame: ted potrebujeme nejakou cast pameti “zamknout” a dovolit jen jednomu procesu k ni pristupovat
      * serializace = bezi jeden za druhym
        + obsluhuje OS
    - muze nastat smrtelne objeti (deadlock) - dva procesy drzi dve ruzne kriticke sekce a vzajemne se blokuji (prvni ceka na sekci 2, druhy na sekci 1)
      * opet musi vyresit OS
      * napr. predepsane poradi, v jakem se muzou kriticke sekce blokovat
    - dalsim resenim je odstraneni sdilenych zdroju - zasilani zprav
      * ale je tam problem synchronizace na urovni zasilani a prijimani zprav
  + planovani
    - jak OS rozhoduje o tom ktery proces se spusti?
    - kazdy OS obsahuje planovac (scheduler)
      * snazi se vybalancovat, aby to bylo vyrovnane (timesharing)
        + napr. casova kvanta (proces bezi dokud se nedostane do stavu zablokovan nebo nevybere pridelene casove kvantum)
      * reaguje na prioritu
        + statisticky rezim - nyni: 2 typy priorit - 1) prideleny pri vzniku a 2) jeho priorita na zaklade uz propocitaneho casu
        + real-time - vzdy bezi proces s nejvyssi prioritou
    - cilem je to, aby si proces nemohl uzurpovat vsechno na dlouhou dobu
* navrh
  + pozadavky: efektivita, robustnost, flexibilita, prenositelnost, kompatibilita, bezpecnost
  + efektivita
    - maximalni vyuziti dostupnych zdroju
    - pouziti jednoduchych a jasnych principu
    - dekompozice navrhu
      * objektove orientovany navrh
        + pozor na to, ze se musi dobre rozlozit na velikost objektu (aby jich nebylo zbytecne moc) → roste rezie
      * agenti
        + nezavisle procesy, ktere neco kontroluji
  + robustnost
    - schopnost uspesne se vzpamatovat po vypadku
    - reseno redundanci
      * vice stejnych komponent, ktere delaji to stejne (jsou jednoduse nahraditelne)
      * napr. tri pocitace → tri vysledky → overeni jestli se rovnaji (pokud ne, tak se bere majoritni vysledek (2 pocitace))
  + flexibilita
    - moznost upravy podle zmenenych potreb
    - casto pouzivana ve vyznamu rozsiritelnost
    - pozadavek, ktery vznikl vyvojem
    - je velmi silnym pozadavkem (casto se prosazuje na ukor efektivity)
  + prenositelnost
    - schopnost abstrahovat od detailu (neresim konkretni hardware - pouzivam jeho “virtualni reprezentaci”
    - tzn. jeden OS muzeme pouzivat pro vice ruznych hardwaru (vzdycky jen pripojime popis konkretnich - ovladace)
    - casto sklonovany je standard!
      * rikaji jak to zarizeni ma vypadat, mit rozhrani a bylo to napric spektrem podobne (v zakladu)
    - opet v rozporu s pozadavkem efektivity
      * protoze specificky hardware lze odladit daleko lepe nez obecny (napr. Apple)
      * potlaceni detailu, ktere nam efektivi vyuziti potlacuji
  + kompatibilita
    - odstineni specifickych detailu
    - opet muze negativne ovlivnit efektivitu
  + bezpecnost
    - snaha o ochranu dat a majetku pred kradezi, zneuzitim ci poskozenim pri soucasnem zachovani pristupu vybranych uzivatelu
    - problemy
      * vetsi naroky na spravu systemu
      * snizuje jednoduchost pouziti
      * klade dodatecne omezeni uzivatele
    - v dnesni dobe zaujima jedno z prvnich mist ve vyvoji
    - napr. Win95 vs NT
      * jednouzivatelsky vs multiuzivatelsky se spravcem
      * diametralne odlisne zabezpeceni
  + externi pozadavky
    - stejny hw a ruzne priority (jedno zelezo na vice cinnosti → jine priority v predchozich bodech)
      * server: stabilita, bezpecnost, propustnost
      * pracovni stanice: snadne ovladani, rozumny vykon ve vsech oblastech
      * specializovana graficka stanice: maximalizace grafickeho vykonu
      * ridici system: pozadavky real-time, robustnost
* klasifikace
  + podle architektury: monoliticky, vrstveny, modularni, mikro-kernel
  + monolititcky
    - puvodni OS
    - abstrakce nepouzivana prilis dovnitr
      * velmi spatne rozliseni funkci (proste jedna velka krava)
    - velmi tezke na spravu (jeden velky balik)
    - spatna rozsiritelnost a spatne udrzovatelny
    - poplatne dobe pomalejsiho vyvoje HW a jeho vysoke ceny
    - puvodni OS z velmi starych PC
    - predpokladalo se, ze pri novem HW dojde k napsani uplne noveho OS
      * casto i pri malych HW zmenach velmi narocne na upravu tohoto OS
  + vrstveny
    - casem se zjistovalo, ze cinnosti jsou velmi obdobne, takze doslo k “vytknuti” funkci
      * obecne veci se napsaly univerzalne, pak uz se delaly jen ty konkretnejsi specificke cinnosti
    - lepsi abstrakce
    - vznik vrstev podle spravy: sprava cpu, pameti, periferii, systemu souboru, …
    - byla v nem daleko lepsi orientace (efektivnejsi provadeni zmen - nemusel parsovat cely “monolit”)
    - popsana komunikace mezi vrstvama
      * tzn. pokud doslo ke zmene vrstvy bez ovlivneni komunikace, tak ostatni mohly zustat beze zmeny
  + modularni
    - misto vrstev moduly
    - jeho vyvoj odpovida vyvoji programovacich jazyku (zacalo se zavadet objektove orientovane proramovani)
    - zapouzdreni (enkapsulace) funkci do modulu
    - komunikace mezi moduly
    - pribuzny OOP
    - lepsi udrzba
    - riziko vzniku “fatware”
      * vznik velmi rozsahlych, kosatych modulu bez jakekoliv funkce
  + jadro (kernel) OS
    - zakladni slozka OS
    - odpovida za alokaci a spravu zdroju, prime ovladani HW, bezpecnost
    - mikrokernel
      * *male je pekne*
      * modularni pristup → ALE male moduly odpovidaji za konkretni operace
      * rada funkci az v uzivatelskem prostoru
        + v jadre zustalo jen co nejmene
        + moduly velmi male a oddelene

tzn. jednodussi zmeny a upravy kernelu

* + - * vysoce flexibilni
        + lze OS upravit podle potreby
        + z jednoho zakladu lze vytvorit hodne OS upravenych na miru
      * posledni (soucasna) faze vyvoje OS
* aplikacni programovaerozhrani (API)
  + definuji zpusob pristupu k OS a dalsim sluzbam
  + sestava se z definici fukcni, datovych struktur a trid
  + predstavuje abstrakci volane sluzby
  + ucel: prenositelnost, snadna sprava kodu
  + dalsi pouziti: preklad mezi sluzbami vysoke a nizke urovne (prevod typu/struktury parametru, prevod mezi zpusoby predavani parametru)
  + udava nam pristup → jestlize spravne zvolim pristup, pak je sluzba daleko lepe vyuzitelna
* periferie z pohledu modularniho OS
  + zpristupneny prostrednictvim API
  + abstrakce
    - lze vymenit kus HW bez vlivu na zpusob pouziti programatory (tzn. nas nezajima konkretni disk...zakladni API pro praci se soubory jsou furt stejne)
  + priznaky a klice pro ovladani specifickych vlastnosti
    - prenositelnost vs. efektivita
  + ovladace na nejnizsi urovni (“nejblize” HSW)
    - kazda periferie ma svuj ovladac
    - specificke jazyky ovladani periferii
    - prace se signaly (napr. zmena stavu periferie)
    - ovladace musi byt SPRAVNE! (je to program !!!) protoze chyba ovladace je spatna a ohrozuje system (obvykle se uz verifikuje)
  + pouzitim API se odrizneme od pouziti specifickych vlastnosti → tzn. musime pouzit nejakou jinou variantu
  + zacleneni ovladace do jadra
    - kooperativni vs. hierarchicke
    - efektivita vs. stabulita
* preruseni
  + OS reaguji na asynchronni udalosti
  + je to mechanismus, jak prerusit vykonavanou praci na zaklade externi priciny
  + vyznam
    - podpora I/O
    - problem v programovem vybaveni
      * neautorizovany pristup
      * nelegalni instrukce nebo operandy
    - pozadavek pocitacem rizene systemu
      * napr. senzory sledujici problemy
    - zasah operatora
    - vypadek HW
  + napr.
    - preruseni od casovace
      * casovac = stara se casove planovani procesu na procesoru
    - preruseni od periferie
      * klavesnice, mys, sitova karta
      * psani na klavesnici
    - preruseni od procesoru
      * chybne operace (napr. deleni nulou)
  + OS stale vyrizuje preruseni (=udalosti)
  + principy
    - prerusi beh aktualniho programu
      * nutno ulozit stav a zapamatovat si misto navratu (odkud se ma spustit)
    - vice zdroju a pricin preruseni
      * nutno rozlisit typ (pricinu) preruseni
      * nutno zapamatovat zdroj preruseni → aby se to dalo vyresit
    - nektere muzeme ignorovat nebo potlacit
  + obsluha preruseni je realizovana v kernelu
    - musi zajistit serializaci (zpracovava jedno po druhe)
    - musi zajistit bezpecnost
    - vyvola tzv. prepnuti kontextu (nasledek preruseni v systemu)
  + 2 kategorie
    - maskovane
      * OS se tvari, ze se nic nestalo (bud docasne nebo trvale)
      * mozna ztrata preruseni/udalosti
    - nemaskovane
      * nelze je maskovat (ignorovat)
      * ma vyssi priroritu (tzn. preruseni se ne vzdy zpracovavaji serializovane)
  + pooling
    - opakovane dotazovani (na stav/udalost)
    - mozna alternativa pro nektera preruseni
      * zamestnava procesor
      * muze zustat v uzivatelskem prostoru
* system souboru
  + zakladni fce:
    - vytvoreni souboru, cteni/psani do/z souboru, odstraneni (smazani) souboru, spusteni souboru (pouze program)
  + ma podporu na urovni OS
  + struktura
    - hierarchicke systemy
      * koren (root)
      * adresar je specialni typ metasouboru, kde adresar drzi informace (kde jsou fyzicky na disku) o souborech, ale nikoliv jejich data
    - databazove systemy
      * soubory jako polozky v db
      * ma bohatsi mnozinu operaci
      * ale je slozitejsi na implementaci → dneska se temer vubec nepouzivaji
  + struktura souboru
    - posloupnost bytu - vnitrni struktura pro OS neznama
    - posloupnost zaznamu
    - strom - kazdy uzel ma vlastni klic
  + unix
    - typy souboru
      * radne - bezne soubory
      * adresare - udrzeni hierarchicke struktury
      * specialni - pristup ke konkretnim zarizeni (/dev/mouse, /proc)
      * blokove - nahodny pristup na zakladni urovni (/dev/hd)
    - pristupove metody
      * sekvencni, nahodny, indexsekvencni (neni v UNIXu)
  + struktura na disku
    - souvisle
      * souvisla posloupnost bloku
        + slozita alokace - musime naalokovat cele misto, tzn. najit na disku souvisle misto minimalne velikosti souboru
        + plytvani mistem (ignorujeme mensi mezery)
    - provazany seznam
      * kazdy blok odkazuje na dalsi
        + muze rust
        + vyssi rezie - pro ukazatel
        + slozity nahodny pristup
        + dlouhe vyhledavani
    - indexove
      * napr. FAT
        + muze byt i fragmentovan
      * tabulka pro vsechny bloky na disku
      * provazany odkazem na dalsi blok daneho souboru
    - inodes
      * v UNIXu
      * vychazi z indexu
      * ruzne urovne tabulek
      * pevna delka tabulky pro kazdy soubor
        + pro delsi soubory jsou alokovany dalsi tabulky
      * flexibilni (pro male soubory minimum)
      * mala rezie
      * pocet neprimych odkazu a velikost tabulek je v prime zavislosti na velikosti souboru
  + volne bloky?
    - v tabulce
      * fat to mel uz “zabudovane”
    - bitovy vektor
      * nejobvyklejsi
      * ekvivalent fat tabulky po bitech - pro kazdy blok na disku jeden bit
    - provazany seznam
      * neni tak nesmyslny jako v pripade dat
      * jen je trosku problem s fragmentaci - hur se hledaji souvisle oblasti
    - nejcasteji zpracovavany FCFS (First Come First Served)
      * opet napomaha fragmentaci - neklade se duraz na souvisle oblasti
  + vyrovnavaci pamet
    - spousta problemu zpusobena latenci (zpozdeni) - signal se siri konecnou rychlosti (chvili trva nez se signal prenese)
      * u vsech komponent
    - snazime se minimalizovat plytvani casem
    - jsou v nejcasteji pouzivane bloky/soubory
    - problem: konzistence pri pristupech/zapisech z vice mist
      * zapisu do pameti, ale co kdyz nekdo chce ta data a precte je z disku?
    - ikdyz program cte treba po bitech, pak disk si do vyrovnavaci pameti nacte daleko vic, aby ta reakce byla rychla
    - pamet neni schopna byt bez elektriny → proto:
      * zakladni typy vyrovnavaci pameti:
        + *write-through* - okamzite po zapisu i na disk (po zapisu do pameti hned take na disk) -- nezrychluje zapis!
        + *write-back* - data na disk asynchronne az po urcite dobe (30s)

# Bezpecnost a ochrana

* obecna ohrozeni:
  + cteni nezanechava stopy - nejsme schopni dohledat, zda-li byl soubor precten nebo ne
  + zapis - smazani/prepsani dat, muzeme pracovat s daty, ktere nejsou relevantni
  + znepristupneni sluzby (denial of service)
* mozne utoky
  + prihlaseni, impersonifikace
    - problem, protoze malokdy mame zakazano cist vlastni data
  + trojsky kun
  + viry
  + socialni inzenyrstvi
    - uhodnuti nebo ziskani heslo
    - vyuziva duvery a naivity lidi
    - mozna dnes nejcastejsi utok vubec
    - dost casto nastava problem s tim, ze se nemeni napr. pristupove udaje na ruzna zarizeni (nekdy zustavaji i defaultni od vyrobce)
    - tochnologie muze pomoci jen do jiste urovne
      * lze napr. pres nutnost koordinovane shody dvou ci vice lidi
        + vice lidi, kteri maji ruzny klic → teprve jejich kombinaci je umoznen pristup
      * kombinace fyzickych nastroju a tajemstvi (karta + PIN - po kradezi karty se jeste musi ziskat PIN - jinak neni k nicemu)
  + vyuziti technickych nedostatku
    - bezpecnosti diry a zadni vratka
    - je mozne minimalizovat programatorskymi praktikami a updatovanim aplikaci
    - existuji nastroje pro automaticke otestovani systemu
  + botnety
    - spojeni napadenych pocitacu do velke site
    - tyto pocitace lze pak vyuzit k dalsim utokum (velky vykon)
      * napr. DDoS
* bezpecnost v navrhu OS
  + zverejneni sifrovacich a souvisejicich algoritmu
    - mnoho lidi stale zastava nazor proti zverejneni sifrovacich algoritmu
      * plati pouze chvili - utocnik muze najit chybu a je problem...zodpovednost lezi pouze na jednom clovekovi/firme
    - pro zverejneni mluvi fakt, ze to nekteri lide cvicne zkousi prorazit a potom to zverejni → vime, ze je to dale nepouzitelne a ze nejsme zabezpeceni
      * podili se na tom vice lidi
  + standardni nastaveni = zadny pristup
    - spravce/uzivatel musi aktivne rozhodnout co komu dovoli
    - v USA se stalo, ze po nainstalovani noveho Windows tak do 7 minut byl zapojenej do botnetu
  + prace s minimalnimi opravnenimi
    - nepracovat jako root, ale jako uzivatel → odolnejsi na chyby
  + pravidelne kontroly a instalace aktualizaci
    - zaplatovani der
    - kontrola napr. sluzeb - upozornuje na nadbytecna opravneni
      * castecne paranoidni
  + jednoduchy a uniformni mechanismus
    - jakekoliv zesloziteni vede k nepochopeni a tim padem vznikaji dalsi chyby
    - jak presvedcit uzivatele o tom pouzivat nejakou bezpecnost??? je to moc slozite, namahave, “zbytecne”
  + urovne opravneni
    - delegace opravneni na konkretni akci
    - system by to mel jednoduse umoznovat (stale problem u Woken)
  + principy jsou univerzalni - plati od vzniku PC...jen se meni hrozby a tim padem na to pada stale vetsi duraz
  + ochrana souboru
    - zakladni opera - cteni, zapis, smazani, prodlouzeni a spusteni souboru
    - zakladni ochrana
      * ruzna pro ruzne operace
      * specifikace kdo co smi
      * ochranne domeny (UNIX)
        + skupina, ktera ma stejna prava
        + staticke vs. dynamicke

staticka - vycet uzivatelu

dynamicka - pravidlo, ktere rika co a jak (napr. jsou v ISu a jsou zapsani v tomto roce v tomto predmetu, patri do tehle skupiny)

* + - * + napr. POSIX v UNIXu: user-group-other (777)

jen jeden z pristupu

* + - rozsireny system: ACL (Access Control List)
      * ke kazdemu souboru pripojen seznam pristupovych opravneni
      * sestava se z usporadanych dvojic (domena, operace)
      * komplexnejsi oproti POSIX zpusobu z UNIXu
      * podporuje Windows i UNIX
      * libovolny pocet zaznamu
      * pozitivni i negativni zaznamy
      * umi dynamickou dedicnost - propagace zmen do podadresaru
    - capability list (CL)
      * usporadani podle domen, nikoliv podle souboru
      * schopnost (capcbility) - prava pristupu patri procesu a ten je muze
        + predavat dalsim procesum
        + modifikovat (degradovat, rozsirit)
        + smazat
      * proces se pri pristupu prokazuje odpovidajici “schopnosti” - capability
        + neni to tak, ze musi byt v nejake skupine, ale napr. to, ze ma cerveny token
      * moznost transferu schopnosti mezi procesy
        + vhodne pro distribuovane systemy
  + vnitrek OS
    - kernel a uzivatelsky prostor
    - oddeleni na HW urovni
    - pristup k pameti
      * kazda stranka nekomu patri - nejakemu procesu
      * znemozni zasah do neexistujici stranky ci stranky, ktere procesu nepatri
      * ochrana
        + mezi procesy a jadrem - nas proces vola jadro a nesmi prepsat jeho data
        + mezi procesy - procesy si navzajem nesmi prepisovat data
        + uvnitr procesu - napr. sahani na nenaalokovanou stranku
    - pouze kernel ma pristup k hardware
      * kontroluje prava pristupu
      * obsluhuje vsechny zarizeni a odpovidajici data deleguje procesum
        + takze kazdy proces dostane jen data, ktere jsou pro nej urcena
      * uzivatelske programy pouzivaji volani kernelu - provede za ne komunikaci s HW a vrati data
    - je velmi dulezita korektnost kernelu!!!!! jinak proces muze dostat neomezena opravneni (ruzne exploity)
* V JEDNODUCHOSTI JE SILA!

# Client-server model

* distribuovane pocitani
  + vyuziti prvku (pocitace) propojenych siti
  + dekompozice ulohy na podulohy
    - paralelni vykonavani poduloh - tzn. vykonavaji podulohy nezavisle na ostatnich pocitacich
    - nektere ulohy je velice slozice dekomponovat → pro usnadneni vznikl client-server
* specialni pripad distribuovaneho pocitani
  + je vice strukturovane
* asymetricky pristup
  + klient posila pozadavek, server zpracuje a odesle odpoved
  + server muze byt pro vice klientu
* client-server
  + client a server jsou uplne nezavisle (vymena dat je pouze konkretni - nesdili vsechno)
  + umoznuje autentizaci (overeni obou stran)
  + duplexni protokol - komunikace na obe strany muze byt provdena nezavisle
    - nevyzaduje synchronizaci komunikace
  + data → poslu na server → server pocita → dostanu odpoved (nejjednodussi pristup - muze byt i mnohem slozitejsi)
  + vlastnosti
    - interoperabilita
      * system i server muzou bezet na rozdilnych platformach
      * napr. prvni program vyzaduje Linux a druhy Windows → dokazeme jednoduse ziskat vysledky obou
    - portabilita
      * data lze distribuovat na vsechny platformy bez nutnosti sjednoceni platforem (napr. webove stranky - funguji “stejne” na ruznych platformach)
      * jsme schopni server pouzivat z ruznych platforem
    - integrace
      * skladani ruznych nastroju/platforem do funkcniho celku (volani ruznych serveru - probiha pouze vymena dat)
    - transparence
      * klient vidi jen sve pripojeni na konkretni server - neni schopny monitorovat dalsi cinnost serveru (tzn. klient nemusi vedet, jak spolupracuji servery...proste to funguje)
    - bezpecnost
      * dovoluje zakomponovat autentizaci a sifrovani
      * muzou vznikat “duveryhodne servery” - nekdo duveryhodny rekne, ze je to ten spravny server (napr. certifikaty) a ze se tedy budu pripojovat na ten spravny
        + napr. internetove bankovnictvi - pripojuji se opravdu na ten spravny server??
        + server muze byt i fyzicky dobre zabezpecen (take vlemi dulezite!!)
  + napr. ssh, telnet, x11, www, afs, nfs, samba, …
  + je velice stary a rozsireny
  + trivrstevny model
    - zjemneni client-server
    - vrstvy
      * data
        + ulozena na serveru
        + napr. databaze
      * logika
        + jak jsou data organizovany, jake maji vztahy, omezeni
        + a co s nima muzeme delat
      * reprezentace na klientovi
        + pohled na data
        + co opravdu na zaklade dotazu dostaneme
    - lze kombinovat sousedni vrstvy a rozdelovat je na klienta a server
      * zalezi na pozadavcich na zabezpeceni
      * **tlusty klient (fat)**
        + snazi se maximum operaci provest lokalne (data - server; logika, reprezentace - klient)
        + znacna spotreba lokalnich zdroju

problem muze byt napr. u mobilnich telefonu - nejsou zatim tak vykonne

* + - * + komplexnejsi aplikace → horsi portabilita
        + z pohledu uzivatele muze byt slozita konfigurace
        + hlavni vyhodou je, ze malo zatezuje server (jen stahnuti dat)

delegovani zateze na klienty → tzn. slabsi server staci i pro hodne uzivatelu

* + - * **tenky klient (thin)**
        + maximum chceme delat na serveru

je daleko snazsi zatizit/pretizit server

* + - * + mensi skalovatelnost (prilis mnoho prace dela server)
        + velmi jednoducha aplikace
        + snadna sprava a prenositelnost
        + uzivateli se jednoduse konfiguruje
        + vyssi naroky na propustnost site

byly by pomale odezvy a aplikace by nebyla tak interaktivni

* + - * + hodi se predevsim pro bezpecne aplikace

vse se pripravi na serveru a uzivatel dostane pouze holy vysledek (tzn. je slozitejsi se do toho naborit)

* + - * nejcasteji nejaky vhodny kompromis (podle urceni aplikace)
* **middleware**
  + stoji nad OS, ale pod aplikaci
  + software zajistujici funkcionalitu distribuovanych systemu
  + propojuje oddele komponenty distribuovaneho systemu
    - napr. distribuovane souborove systemy
  + dovoluje aplikacim pracovat ve vice abstraktni vrstve
    - server poskytne stejnou funkcionalitu jako lokalni zdroj (napr. souborovy system - muzu mazat, kopirovat, … a nerozlisuju, zda-li je to vzdaleny FS nebo lokalni)
  + muze byt specializovany (poskytuje jen jednu vec) nebo byt univerzalni (poskytuje ruzne veci a programatori nemusi kombinovat vice middlewaru)
  + protokoly
    - RPC - remote procedure call
      * velmi jednoduche
      * neumoznuje autentizaci a sifrovani
    - DCE - distributed computing environment
      * rodina middlewaru, ktera zahrnovala RPC, autentizaci, vzdaleny pristup k FS, adresarove sluzby (spojovani informaci o uzivatelich)
    - CORBA, OGSA - open grid service architecture
      * komplikovanejsi distribuovane objektove sluzby
  + CORBA
    - zakladem je vrstva ORB, ktera zprostredkovava komunikaci mezi komponentama (middleware pro middleware)
    - komponenty
      * rozhrani (retezce)
        + umoznuji volani procedur mezi klientem a serverem
      * pojmenovani (naming service)
        + genericke pojmenovani toho co chceme a teprve pri potrebe se zacne vyhledavat, kde se fyzicky nachazi takto pojmenovana komponenta
      * obchodni sluzba (trader)
        + vyhledavani vhodneho serveru
      * …
  + webove sluzby
    - vyuzivaji se standardni protokoly od W3C
    - urceny pro interakci mezi pocitaci pres sit
    - protokoly: HTTP, HTTPS
    - WSDL (web service description language)
      * popsani rozhrani sluzby
    - SOAP (simple object access protocol)
      * vzajemna vymena zprav
    - XML (eXtensible markup language)
      * znackovaci jazyk pouzivany pro popis objektu a vlastni komunikaci

# Mobilni systemy

* potencialni vyuziti distribuovaneho prostredi
* zajimaji nas mala zarizeni, ktere pripojujeme do site
* inherentne distribuovane
  + primerene mala zarizeni
  + trvale nebo obcas pripojena k siti
  + omezena procesni kapacita
* pouziva se i client-server a peer-to-peer komunikace
  + client-server se lepe prizpusobuje vykonu (umoznuje lepsi rozlozeni toho co dela server a co klient - dokonce i dynamicky)
  + peer-to-peer
    - obe zarizeni muzou byt klienti i servery → tzn. spatne se deleguje narocnost vypoctu → min idealni pro mobilni prostredi
    - ocekava, ze client i server jsou rovnocenne
* velmi vyznamny problem: jak osetrit praci v odpojenem stavu?
  + jak resit nahodne pady?
  + zarizeni by melo jit pouzivat (alespon castecne) i bez pripojeni k siti
    - napr. cachovani stranek, dokumentu...muzu i bez staleho pripojeni pracovat se “zamrzlou” verzi dokumentu
  + mel by s tim dopredu pocitat system
  + otazka vyvazovani vyuziti pripojeni a vykonu pristroje
* postupne roste rychlost zarizeni
* zvysuje se dostupnost a kvalita siti
* zacinaji se aplikace nabizet pomoci sdileni (musi byt vzdalene dostupna)
* senzorove site
  + mala zarizeni, ktera sleduji stav okoli
  + detekce pozaru, tlak, vlhkost, …
  + komunikacni zatez
    - omezena kapacita baterie
    - jakakoliv zatez musi byt velmi efektivni na spotrebu energie (napr. pripojeni do site se deje pouze nekdy, …)
      * napr. algoritmy jsou hodnocene podle “ceny” ve wattech (nikoliv poctem instrukci) - uplne odlisny pohled na programovani
    - senzory komunikuji mezi sebou (postupne si predavaji informace) → nutne protokoly garantujici primerenou konektivitu, ale nezatezuji moc baterii
  + bezpecnost
    - utocnik se muze senzoru fyzicky zmocnit → riziko posilani spatnych dat
    - kompromitace senzoru nesmi ohrozit celou sit
      * je nutne pocitat s chybovosti senzoru

# Autentizace a autorizace

* autentizace proces, kterym se prokaze, ze ja jsem ja
* autorizace je proces, ktery nam rekne, co muzeme delat (opravneni ke sluzbam/zdrojum)
* delegace je prokazani, ze muzu vystupovat za nekoho jineho
  + typicky plna moc
* zpusob autentizace
  + neco co mam
  + neco co jsem
  + neco co vim
  + → ruzne miry spolehlivosti a vhodnost pouziti
* autorizace nasleduje po autentizaci - rekne nam, co muzeme delat
* velmi dulezita je kryptografie
  + snaha o ochraneni komunikace proti utocnikum/neopravnenym osobam
  + dalsi pozadavky na zpravu: integrita (zprava nebyla zmenena), autenticita (zda-li je odesilat opravdu spravny), non-repudibality (neodmitnutelnost - chci mit dukaz, ze zprava dorazila)
  + sifrovani zajistuje pouze necitelnost zprav
* sifry
  + symetricka sifra
    - stejny klic pro sifrovani a desifrovani
    - problem distribuce klice (pomoci klice lze hned rozsifrovat)
    - pokud je klice delsi nez zprava, pak je to nerozbitelne (ale v realu jsou obvykle zpravy delsi nez klice)
    - aktualne nejpouzivanejsi algoritmus: AES
    - pouziti v autentizaci
      * neposilam rovnou tajemstvi (posilat heslo je riskantni)
      * jedna strana zvoli nahodne cislo, zasifruje a posle
      * druha desifruje, provede dohodnutou operace, znovu zasifruje a posle zpet
      * prijemce desifruje a zkontroluje vysledek
      * tento proces je zakladem *Challenge-Response protokolu*
    - rizika/problemy
      * distribuce hesla
      * kompromitace hesla
      * vicebodova komunikace
  + asymetricka sifra
    - 2 klice - verejny a soukromy
    - oba mohou byt pouzity pro sifrovani i desifrovani, ale komplementarne
      * zprava zasifrovana soukromym klicem je desifrovatelna pouze verejnym a naopak
    - jednoduse distribuovatelny - znalost verejneho klice
    - problem jak prokazat, komu patri dany verejny klic
    - pouziva jednosmerne funkce
    - pouziti v autentizaci
      * jedna strana zvoli nahodne cislo a zasifruje verejnym klicem druhe
      * druha strana desifruje soukromym, provede operaci a zasifruje verejnym prvni strany
      * prvni strana desifruje svym soukromym klicem a overi
    - rizika/problemy
      * autenticita verejnych klicu
      * nevhodne pro sifrovani dlouhych zprav
    - pomalejsi nez symetricke
* certifikacni autorita
  + potvrzuje platnost verejnych klicu
  + overuje, kdo je vlastnik soukromeho klice k urcitemu verejnemu
  + vydava certifikat, ktery potvrzuje tuto vazbu, ktery sama podepise
  + problem: jak verit klicum certifikacnim autorit?
    - a jak verit korektnosti autorit?
* digitalni podpis
  + vytvori se hash zpravy (jedinecny otisk pro konkretni zpravu) → ten se zasifruje soukromym klicem (“podepise”) a pripoji se k souboru → odesle se
    - tzn. pokud kdokoliv ten dokument opet prozene hashovacim algoritmem a vyjde mu to stejne co me, pak je dokument “porad stejny”
  + vyuziti asymetricke kryptografie
  + z hashe nejde rekonstruovat puvodni zprava
  + drive se pouzival MD5 (ten uz neni uplne bezpecny), ted SHA-1
  + hash se pouziva, protoze sifrovani cele zpravy je vypocetne velmi narocne (zvlaste, je-li delsi)
  + princip pouzitelny i na garanci integrity a autenticity zpravy
* delegace
  + potrebujeme poverit nejakou entitu, aby mohla jednat nasim jmenem
  + nejde resit v symetricke kryptografii
  + naivni pristupy (nekorektni a nebezpecne)
    - sdelime sdilene tajemstvi
    - sverime soukromy klic
  + v realu se pouziva asymetricke kryptografie:
    - vydame novy certifikat, ktery podepiseme
      * entita se prokazuje timto novym (ma jeho vlastni soukromy klic)
      * druha strana vidi nas podpis pod delegaci a proto akceptujeme
  + algoritmy dnes dokazi dobre delegovat do 4 urovne (potom uz maji problemy a chovaji se nekorektne)
* duveryhodnost
  + castejsi pouziti klice zvysujeme pravdepodobnost odchyceni/zneuziti
  + nektere systemy nemusi pouzivat dostatecne spolehlive systemy overeni
  + napr. mame primarni a sekundarni heslo v ISu
  + take treba zalezi na tom v jakem prostredi se prihlasujeme (napr. rozdilna bezpecnost meho PC a PC v kavarne)
  + zalezi na tom jak ruzne druhe strany povazujeme za ruzne duveryhodne
    - snazime se proto pouzivat ruzne autentizacni/komunikacni mechanismy
    - chranime sdilena tajemstvi
  + dalsi pristupy
    - digitalni karty
    - poskytovani identit, federace identit
  + zalezi i na osobnim citeni (jak moc si cenime takhle zabezpecenych informaci?)
  + problem: idealne mit pro kazdy system jine heslo? to neni realne...
    - tzn. vhodny kompromis pouze s nekolika urovnemi (kazda uroven = jedno heslo)
* jak zasifrovat dlouhou zpravu?
  + asi symetrickym klicem, protoze je daleko rychlejsi
* jak sdelime klic druhe strane?
  + nejlepe vyuzitim asymetricke kryptografie
  + verejnym klicem druhe strany zasifrujeme symetricky klic a prilozime ke zprave
* *=> idelani je kombinace obou zpusobu*

# Grafika

* oblasti:
  + jak obraz vytvorit a reprezentovat?
  + jak obraz manipulovat a zobrazit?
    - zvetsovani?
  + jak napodobit realitu?
    - jak napodobit to co vnima nase oko?
    - najit prevod do obrazu tak, aby neslo rozpoznat digitalne vytvoreny a realne vyfoceny obraz/video
      * pouziti fyzikalnich modelu
  + jak to rychle vypocitat?
    - GPU je pocetne nejvykonnejsi soucast PC - protoze graficka zpracovani vyzaduji obrovsky numericky vykon
    - snaha o tom jak to rychle pocitat
* **analyza obrazu**
  + obrazova data → modely
  + extrakce informaci z obrazovych dat
  + carove kody, detekce pohybu, identifikace objektu/osob, ...
  + napr. Kinect - dokaze pomoci kamer snimat pohyby tela a prenaset je do hry
  + ve filmech se pouziva snimani pohybu a jeho nasledne preneseni do postavy (tzn. fyzika se nepocita *ab initio*, ale na zaklade realnych osob)
    - *ab initio* pristup jeste nejsme schopni udelat
* **synteza obrazu**
  + modely → obrazova data
  + vytvoreni obrazovych dat na zaklade datovych modelu
  + uprava fotografii, realisticke zobrazeni 3D scen, vizualizace dat, …
  + resi i problemy jak vytvorit objekty
    - napr. jak vytvorit strom? - existuji L-systemy (fraktaly), ktere dokazi vytvorit ruzne verohodne stromy
  + venuje se i 3D - ve vsech aspektech
* **vizualizace**
  + cil: zobrazeni komplexnich dat uchopitelnou formou
    - plati pro simulovana data i vysledky mereni
    - snadnejsi pruchod daty a jejich analyza
    - usnadnuje porozumeni obejnym/komplexnim datum
  + jak vhodne zobrazit data a tim usnadnit nalezeni “informaci” v nich
    - nas mozek zpracovava obrazovy vstup a umi v nem najit spoustu vzoru a opakujicich se sekvenci (napr. grafy - v originalnich tabulkach bychom rozhodne nenasli sekvence)
  + hledanim nasima ocima (resp. mozkem) je casto efektivnejsi nez nalezt algoritmus, ktery by dokazal to stejne (UI jeste neni na odpovidajici urovni) → je dulezite data spravne vizualizovat
    - ani zatim nemame dostatecne vykonny HW
  + aplikace
    - vysledky vedeckych experimentu a simulaci
    - medicina (sonograf, magneticka rezonance) - pro to existuji i “surova data” (pouze cisla)
    - strojirenstvi, defektoskopie (vyhledavani poruch/mikrozlomu), chemie, biologie, geologie, …
* graficky subsystem (obrazek viz. slajd)
* predpoklada zobrazeni pres rastr (dvojrozmerna mrizka o presne danych rozmerech, kde kazdy bod - prusecik - umime rozsvitit, dat jim jas a barvu) → obraz ze zmeti bodu se v nasem mozku spoji a vytvori celkovy obraz
  + zobrazime body a my vnimame plochy
* graficky akcelerator (spocita nam tu dvourozmernou obrazovou mapu) → obrazova pamet → video radic → monitor
* graficke procesory musi byt velmi vykonne, protoze rastry jsou velike
  + ale jsou prizpusobene pouze na takove operace
  + pouziva docela malo pameti
* pamet v GPU slouzi predevsim jako cache na predpocitane bitmapy
* fyzikalni podstata barvy
  + svetlo je elektromagneticke vlneni charakterizovane vlnovou delkou a intenzitou
  + viditelne spektrum: 400 nm (fialova) - 700 nm (cervena)
    - existuje i “neviditelne” spektrum (napr. UV, infracerveny)
  + bile svetlo = obsahuje vsechny vlnove delky ( → pomoci hrnolu je lze rozdelit)
* oko
  + oko si mnoho veci dopocitava samo
  + lze ho osalit - napr. mizeni obejktu (na jednom zaberu objekt je, na druhe neni)
  + pro realisticky obrazek: oko vzdy jde v ruku v ruce s mozkem
* barevne prostory
  + oko rozpoznava velmi velke mnozstvi barev (mame hodne receptoru)
  + na kazdem zarizeni lze zobrazit jen urcite barvy
  + **barevny gamut**: mnozina barev, kterymi dane zarizeni disponuje
  + barvy monitoru by mely byt maximalne podobne nasemu oku, aby byly barvy “prirozene”
  + kazdy monitor ma jiny gamut - tzn. stejny obrazek na ruznych monitorech se muze dost lisit
* kodovani barev
  + pro zobrazeni barev je nejlepsi vyuzit vlastnost naseho oka a barvy “skladat” - kombinace ruznych monochromatickych kanalu, kde se pro kazdou barvu meni jejich intenzitu
  + **RGB (red-green-blue)**
    - aditivni: scita barvy k bile
    - odpovida skladani svetla → monitory
    - pouziva se tam, kde “se sviti”
  + **CMY (cyan-magenta-yellow)**
    - subtraktivni: odecita barvy od bile
    - odpovida michani barev → inkoust
    - cim vic barev → tim vic cernejsi
* opticke iluze
  + jak “podvest” nase oko
  + lidske oko “osciluje” mezi ruznymi variantami/vysvetlenimi obrazku
* rastrovy obraz
  + obraz je 2D pole pixelu
  + barva kazdeho pixelu je definovana bity (**b**) - tzv. barevna hloubka (kolik ma kazda barva bitu)
    - 1 bit - cernobily obraz
    - barevny obraz: 8, 15, 16, 24 (True Collor), …, az 96 bitu
* technologie displeju
  + CRT - tri svazky elektronu jsou urychlovany a cileny na luminoforovou vrstvu s RGB oblastmi
  + LCD - organicke molekuly ulozene mezi dvema polarizacnimi filtry s kolmymi osami polarizace
    - v klidove poloze polarizuji svetlo o 90 stupnu a umoznuje jeho pruchod
    - v excitovane poloze nepolarizuji a pixel se jevi jako nerozsviceny
    - nevydava svetlo: vyzaduje podsviceni ci reflexni vrstvu
  + plazma - plyn uzavreny v malych bunkach (3 na pixel) je excitovan el. polem a vydava UV zareni → UV zareni dopada na fosfor uvnitr bunky a ten vyda viditelne svetlo
    - plazmove, protoze plyn dostavame do formy plazmy
  + OLED - nekolik vrstev organickeho materialu ulozenych mezi anodou a katodou → pri pruchodu el. proudu organickym materialem dochazi k emisi viditelneho svetla
    - aktivni zdroj svetla (nepotrebuje podsviceni), ohebne
    - je velmi ohebny
  + dotykove displeje
    - spojeni obrazoveho vystupu a hmatoveho vstupu
* rastrova konverze usecek
  + cil: prevedeni spojite usecky do rastrove reprezentace
    - usecky muzou zasahovat do vice pixelu
    - resenim je pocitani nejblizsiho pixelu v ose y
      * vypocet pomoci round() v kazdem kroku je neefektivni
      * pouziva se inkrementalni vypocet: Bresenhamuv algoritmus
* vypln ploch
  + cil: obarveni vsech pixelu v dane oblasti
  + mozne definice oblasti:
    - vsechny pixely dane barvy
    - vsechny pixely v dane vzdalenosti
    - oblast definovana polygonem
  + definice sousedniho pixelu
    - 4 smerna: spolecna hrana
    - 8 smerna: spolecne hrana ci vrchol
  + obarvovani:
    - zaplavove
      * vyberu jeden pixel v oblasti a pak vsechny ostatni obarvuju rekurzivne
    - radkove
      * rekurze probiha po sousednich radcich, ne pixelech
      * vyrazne efektivnejsi
  + vypln polygovu
    - najdi pruseciky radky s polygonem
    - serad podle polohy na ose x
    - vybarvi sude useky
    - samozrejme jsou tam sporne pixely - zalezi na nasi implementaci/algoritmu
* antialiasing
  + prevodem spojiteho obrazu na diskretni rastrovou reprezentaci vznikaji chyby
    - ztrata detailu
    - vznik nezadoucich artefaktu
    - rozpad tvaru
    - reseni:
      * zvysene rozliseni
      * predfiltrovani
        + aplikuje se behem rasterizace
        + kazdemu pixelu je nastavena intenzita pomerne k velikosti plochy, kterou je zakryvan rasterovanym objektem

sice dojde k mirnemu rozostreni, ale zaroven je to realistictejsi

* + - * + nekdy tezke rict, ktery vysledek je lepsi
      * postfiltrovani
* modely a modelovani
  + cil: popsat co je na obraze
  + ze zakladnich primitiv se skladaji komplexni tvary
  + 2D vektorova grafika
    - usecka, krivka, elipsa/kruznice, mnohouhelnik, …
  + 3D popis povrchu
    - 2D objekty s obsahem, parametricke plochy, spojovani platu, …
    - nemusime videt “dovnitr” objektu - staci pouze popis povrchu
  + parametricke krivky
    - napr. Bezierovy krivky
      * nemusime si pamatovat vsechny body, ale staci pouze 3 body a zbytek se jednoduse dopocita
  + parametricke plochy
    - umoznuji popis hladkych zakrivenych povrchu
    - vhodne pro prumyslovy design
    - moznosti definice
      * okrajovymi krivkami
      * polygonovou siti
    - staci manipulace s malo body, ale presto mame zachovanou spojitost povrchu
  + polygonove modely
    - tvar je slozen s konvexnich 2D primitiv
    - snadne vykresleni
    - techniky uprav
      * tazeni (extrudovani) povrchu
      * rotace profilu kolem osy
      * zjemneni a deformace site
      * kostruktivni geometrie teles
  + konstruktivni geometrie teles
    - komplexni objekty jsou z jednodussich vytvareny pomoci boolskych operaci
      * sjednoceni
      * prunik
      * rozdil
  + objemove modelovani
    - prostor uniformne rozdelen na *voxely* (mala trojdimenzionalni kosticka)
    - cele teleso vyskladame z malych kosticek
    - voxely maji ruznou barvu, prusvitnost, fyzikalni vlastnosti, …
    - lze potom telesem delat jakykoliv “rez”
    - aplikace:
      * zobrazeni medicinskych dat
      * objem zadavan po rezech
      * moznost selektivniho zobrazeni
    - “libovolny nahled na libovolne misto z libovolneho uhlu”
  + renderovani
    - cil: vytvoreni obrazu na zaklade modelu
    - prevedeni modelu do rastrove podoby
    - popis sceny
      * geometrie objektu
      * osvetleni - kde je zdroj svetla?
        + popis zdroju svetla, ruzne modely sireni svetla
      * textury - jaka je barva a povrch objektu?
        + urcuje opticke vlastnosti povrchu
        + pridava detailni zmeny geometrie (normalove mapy - vystupky)
        + rastrove

rastrovy obraz je namapovan (natazen) na povrch

vysledek je ovlivnen rozlisenim textury a pouzitou interpolaci

* + - * + proceduralni

vlastnosti pixelu jsou zadany funkci

vyzaduje programovatelny HW

dobre skaluje na vysledne rozliseni obrazu

* + - * smer pohledu
      * stinovani
        + pomoci stinu vnimame hloubku, tvary
      * raytracing
        + sleduje cestu paprsku z oka do zdroju svetla
        + dany stupen odrazu
        + umoznuje stinovani, leskle povrchy, …
        + distribuovane sledovani paprsku

kazdy paprsek nahrazen svazkem paprsku

vysledkem je prumer ziskanych hodnot

umoznuje hloubku ostrosti, mekke stiny, …

* + - renderovani na GPU
      * renderovani v realnem case
        + nizsi naroky na kvalitu - dulezitejsi je rychlost
        + rasterizace misto raytracingu

geometrie transformovana do 2D

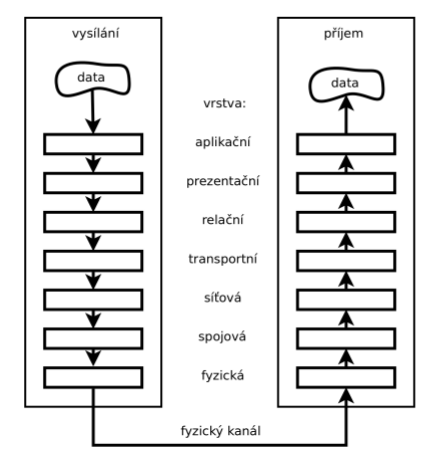
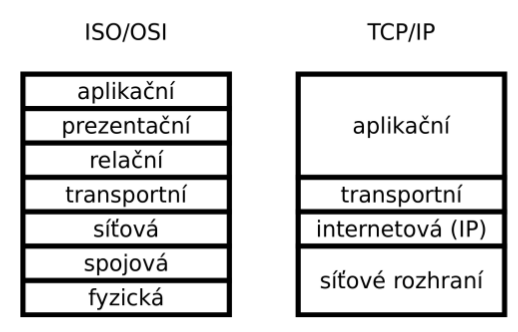
urceny viditelne trojuhleniky

prevedeni na pole pixelu

* + - * programovatelne GPU
        + novejsi GPU umoznuji i jine vypocty
        + komplexnejsi per-pixel efekty (bump mapy, shadery, …)
        + mozne vyuziti i mimo rasterizaci

napr. vedecke vypocty

# Pocitacove site

* skupina HW prostredku a pocitacu propojenych komunikacnimi kanaly, ktere umoznuji sdileni informaci a zdroju
* mozna ucely
  + komunikace uzivatelu
  + sdileni hardwarovych zdroju
  + sdileni dat a informaci
  + poskytovani softwarovych sluzeb
* zakladni vlastnosti
  + doruceni dat spravnemu prijemci
  + spravnost doruceni (integrita dat)
  + vcasnost doruceni
* historie
  + nutnost poradnych grafovych algoritmu, aby bylo schopno data efektivne prenaset
    - napr. Djiskra, Bellman-Ford, ...
    - narozdil od spojovane site, kde je jasne kdo je odesilatel a kdo prijemce
  + sit je decentralizovana (data se tam pohybuje “nahodne”)
  + k rozvoji prispela studena valka
    - proto to bylo decentralizovane (aby nebylo jasne kam utocit)
* idealni site
  + transparentni pro uzivatele/aplikace
  + neomezena propustnost
  + zadne ztraty dat
  + zadne zpozdeni
  + zachovavaji poradi paketu
  + data nemohou byt poskozena
* skutecne site
  + maji vnitrni strukturu, ktera ovlivnuje doruceni dat
  + omezena propustnost
  + dochazi ke ztratam dat
  + data se variabilne zpozduji
  + poradi paketu neni garantovano
  + data mohou byt poskozena
* pozadovane vlastnosti
  + efektivita - efektivni/maximalni vyuziti dostupne prenosove kapacity
  + spravedlnost - stejna priorita pristupu vsech uzivatelu ke vsem datovym tokum
  + decentralizovana sprava
  + rychla adaptace na novy stav topologie site
  + spolehlivost
  + rizeni toku dat - ochrana proti zahlceni
* zakladni pristupy
  + spojovane site (prepinani okruhu)
    - komunikace ve dvou fazich: navazani spojeni → prenos dat
    - nejdrive vytvorime okruh a pak po nem posilame data
    - spojeni je udrzovano behem cele komunikace
      * sit musi udrzovat informace o vystavenych spojenich
    - “abstrakce dratu”
    - nejsou decentralizovane
    - snadne zaruceni kvality sluzeb
    - napr. analogove telefonni site
  + nespojovane site (prepinani paketu)
    - data rozdelena vysilajici stranou na male pakety (datagramy)
    - kazdy paket prochazi siti samostatne
    - pakety muzou prichazet v ruznem poradi
    - prijimajici strana pakety opet slozi do puvodni podoby
    - dnesni internet
    - hur se monitoruje - pakety se stejnou adresou nemusi projit stejnymi uzly → ohrozena kvalita sluzeb
    - je robustnejsi
* komunikacni protokoly
  + motivace: forma komunikace/domluvy musi byt predem znama vsem zucastnenym stranam
    - internet je velmi heterogenni: existuje spousta ruznych HW od ruznych HW → bylo potreba stanovit jasna pravidla jak se bude komunikovat
  + urcuje co je predmetem komunikace, jak dana komunikace probiha a kdy probiha
  + definuje
    - syntax - strukturu/format zasilanych dat
    - semantiku - funkcni vyznam zasilanych dat
    - casovani - kdy je treba zaslat kterou zpravu
* ISO/OSI model
  + 7 vrstev, kazda zodpovida za konkretni funkcionalitu
  + kazda vrstva komunikuje pouze se sousedicimi vrstvami
    - vyuziva sluzeb nizsich vrstev k poskytovani svych sluzeb vyssim vrstvam
    - funkcionalita je izolovana v ramci vrstvy
      * neni moc efektivni → existuji zkratky mezi vrstvami
  + logicky se komunikace odehrava mezi stejnymi vrstvami komunikujicich stran, fyzicky prochazi vsemi nizsimi vrstvami
  + vrstvy popisuji pouze abstrakci, skutecne implementace se mohou odlisovat
* TCP/IP
  + pouze 4 vrstvy
  + fyzicka vrstva
    - propojene 2 body mediem (vodeny - draty, optika; nevodeny - wifi)
      * neni potreba adresace
    - resi se prevod analogovych a digitalnich signalu
      * nektera media neumi analogovy nebo digitalni
  + linkova vrstva
    - potreba adresace: N-bodu
      * pouziti MAC adres (adresy sitovych rozhrani)
    - stavba lokalni pocitacove site
    - ruzne protokoly pro to, kdo a kdy muze vysilat
  + sitova vrstva - IP (internet protocol)
    - cil: propojovani lokalnich siti do velkych, komplexnich siti (internet)
    - odpovida za dopravu dat mezi uzly
    - nespojovana komunikace: zakladni jednotkou prenosu je paket
    - zajistuji smerovani paketu v siti
    - jednoznacna identifikace (adresace) kazdeho sitoveho rozhrani
      * kdyz mam 3 sitova rozhrani, tak kazde z nich musi unikatne identifikovano
    - metody zakladniho monitoringu site
    - MTU - rika, jak velky balicek muze projit linkou
      * kdyz je vetsi → dojde k jeho fragmentaci (rozdeleni) a zaslani samostatnych casti (IPv4)
      * kdyz je vetsi → je balicek zahozen a notifikovan odesilatel (musi ho sam fragmentovat) (IPv6)
      * fragmentace zatezuje smerovac → snizuje propustnost smerovace → proto v IPv6 zahrnuto zahazovani
    - IP adresa
      * jednoznacne urceni uzlu pomoci ciselne adresy
      * adresa IPv4 ma 32 bitu, IPv6 128 bitu
      * typy:
        + individualni (unicast)

komunikuji 2 body

* + - * + broadcast (pouze v IPv4 - v IPv6 nahrazena multicastem)

slouzi k zaslani dat vsem uzlum v dane LAN

* + - * + skupinove (multicast)

data smerovana prijemcum, kteri o ne projevili zajem

* + - * + enicast (pouze IPv6)

data jsou dorucena jednomu clenu skupiny

* + - domenova jmena
      * IP adresy nevhodne pro lidi → vznikla jmenna sluzba - prevod IP adres a jmen
      * DNS (domain name service)
        + zprava hierarchicka (aisa.fi.muni.cz)
        + lokalni/korenove DNS servery

jejich usporadani je hierarchicke

* + - * + ruzne domeny (pro staty, univerzalni, …)
    - IPv6
      * rychle dochazely IPv4 adresy → motivace pro vznik noveho protokolu
      * opravuje nejake chyby IPv4
      * na kazdeho cloveka existuje 2^5 IPv6 adres
      * vlastnosti
        + vetsi adresni prostor
        + odstranuje nutnost prepocitani kontrolniho souctu pro kazdy pruchod smerovacem
        + hlavicka konstantni delky
        + rozsiritelny - rozsirujici hlavicky
        + podpora prenosu v realnem case - prioritizace provozu

lze oznacit prioritni pakety

* + - * + je schopen pracovat s toky
        + podpora zabezpeceni prenosu - autentizace, sifrovani provozu, …

IPSec (zatim doporuceny)

* + - * + podpora mobility pomoci domacich agentu

predpokladame, ze kazde zarizeni je nekde “doma” → pokud nekdo posle na moji “domovskou adresu” dopis, tak nejaky domaci agent vi, kde se prave nachazim → dopis me preposle

* + - * + podpora autokonfigurace
    - prenos paketu
      * cil: dopravit paket skrz sit od vysilace k prijimaci
        + v podstate grafovy problem
        + linky jsou pasivni
        + algoritmy nalezeni cesty mezi dvema uzly grafu
        + uzly znaji cestu (explicitne ci implicitne)
        + podle cilove adresy rozhoduji uzly kam poslat paket dale
        + dve faze

nalezeni smerovacich tabulek (uzly mezi sebou komunikuji)

2 pristupy

o svych vsech znalostech posilam sousedum

vsem posilam znalosti o svych sousedech

podle toho si uzly mapuji topologii site a diky tomu jsou schopni nalezt vhodnou cestu

vlastni zasilani datovych paketu

zasilaji po ceste vybrane ze smerovacich tabulek

* + - smerovaci schemata
      * **distribuovane** x centralizovane
        + centralizovane - stav site se posila do centra, ktere spocte tabulky a zasila je zpet uzlum, v internetu naprosto nepouzitelne
        + distribuovane - vzajemna kooperace uzlu pri vytvareni tabulek
      * **“krok za krokem”** x zdrojove
      * **deterministricke** x stochasticke
        + stochasticke - pravdepodobnostni algoritmy (na 30% pujde tudy a na 70% tudy)
        + deterministicke - je presne definovano kudy pujde
      * **jednocestne** x vicecestne
      * **dynamicke** x staticke
        + staticke - manualne generovane tabulky, jsou definovany od zacatku, pouzitelne pro male site, vhodne pro statickou topologii
        + dynamicke - tabulky aktualizovany v reakci na zmeny v topologii, velmi robustni, lze nezavisle a automaticky mapovat sit
      * vzdy **prvni** protokoly jsou pouzivane v internetu
  + transportni vrstva
    - zavadi pojem portu
      * na jednom uzlu muze bezet vice aplikaci na jednom stroji → nutna jemnejsi adresace
      * port = adresa poskytovane sluzby
    - UDP protokol
      * pouze nezaruceny prenos
      * datagramy jsou zasilany do site bez dalsi kontroly ( → neni zadna garance ani kontrola doruceni)
      * je velmi rychly
        + pouziva ho napr. skype (potrebuje vysokou propustnost → TCP by to strasne zpomalilo)
    - TCP protokol
      * stavi “spojovanou” sluzbu nad IP
        + inicializace spojeni
        + garantovany prenos (lze overit, zda-li paket dosel spravne)
      * je schopny reagovat na zahlceni (zpomali a znovu postupne “pridava”)
      * hlavni prenosovy protokol na internetu
      * ma obrovskou rezii (pry az 40% paketu v internetu je pro “potvrzeni” dosleho paketu)
      * je pomaly
    - existuji ruzne varianty obou protokolu (lze najit ten “nejvhodnejsi” pro nase potreby)
  + relacni vrstva
    - relace (tez dialog)
      * spojeni mezi dvema koncovymi ucastniky na urovni vyssi nez je vrstva transportni
      * castecne sem zasahuje TCP (ale radi se do transportni)
      * analogie telefonniho hovoru
        + vezmu sluchatko a vytocim cislo → vytvorim spojeni
        + pak muzu komunikovate
    - jedno relacni spojeni muze zajistovat vice relaci
    - jeno transportni spojeni muze zajistovat vice relacnich spojeni
    - sluzby
      * rizeni dialogu mezi koncovymi ucastniky
        + plne duplexni - tow way simultaneous
        + poloduplexni - two way alternate

pouziva “pesek” token - vzdy jen jeden uzel prenasi data

* + - * + simplexni - simplex alternate
      * checkpointing synchronizace
        + mame prijemce, ktery nabizi sluzbu
        + pokud dojde k poruse sluzby, tak muze prijemce nabufferovat cast dat
        + nekdy je ale potreba se vratit “zpet” a cast dat poslat znovu (po naplneni bufferu se zahazuje)
        + mechanismus kontrolnich bodu - prijemce muze vyzadat data z kontrolnich bodu

hlavni a vedlejsi

* + - je samostatne pouze v ISO/OSI (v TCP/IP je soucast aplikacni vrstvy)
    - protokoly: SSL, SDP, RPC, …
  + prezentacni vrstva
    - ukol: konverze prenasenych dat do jednotneho formatu
    - mame ruzne architektury, vyrobce, kodovani, uchovani cisel, little-endian, big-endian, … → sit je heterogenni
      * → prezentacni vrstva nam pomaha prelozit data z ruznych architektur → data vypadaji vsude stejne (jednotna interpretace dat)
      * prevod do mezitvaru (abstract syntax notations ASN 1)
    - dalsi sluzby
      * sifrovani
      * komprese
    - v TCP/IP modelu to spada do vrstvy aplikacni ( → kazda aplikace si to ma zarizovat sama)
  + aplikacni vrstva
    - sidli zde aplikace
    - kvuli teto vrstve se “snazi” vrstvy pod ni → je to zakonceni celeho procesu → dava smysl cele siti
    - aplikace lze rozclenit podle kriterii
      * pro ucel: elektronicka posta, web, …
    - protokoly: HTTP, SMTP, FTP, DNS, RTP, RTCP…
      * definuji formu komunikace mezi aplikacemi
      * definuji typy zprav, syntaxi zpravy, semantiku, casovani
      * stejna kriteria jako na ostatni protokoly
    - DNS
      * jmenna sluzba
      * zajistuje preklad domenovych jmen na IP adresy a zpet
        + reverzni domeny
      * byla vytvorena pro “lidi” - nejsme schopni si pamatovat tolik cisel
      * drive byl jen “globalni” host soubory → s rostoucim poctem domen a stroju byly soubory obrovske a spatne se s nimi pracovalo a propagovaly se → proto byly vytvoreny jmenne prostory
        + plochy jmenny prostor - zadna vnitrni struktura

s roustoucim systemem se s nim spatne pracuje

* + - * + hierarchicky jmenny prostor

jmena maji “strukturu”: aisa → fi → muni → cz (opacne to propada pres DNS servery)

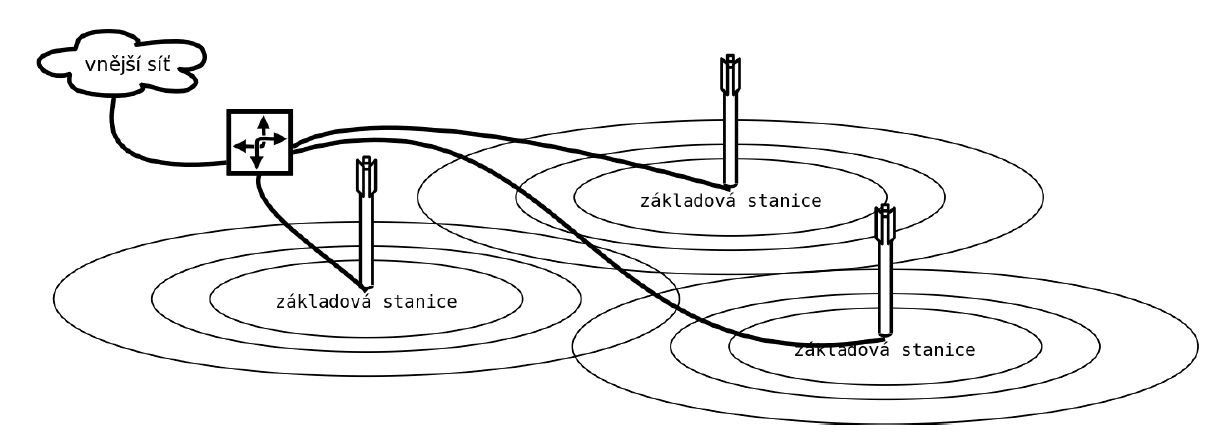
lze decentralizovat spravu

vice nezavislych DNS serveru

normalni stromova struktura

maximalni pocet urovni je 128 (ale nevyuziva se)

* + - * kazdy uzel je charakterizovan jednim stringem (max 63 znaku)
      * 13 zakladnich korenu (charakterizovan prazdnym retezcem)
        + je jich vice proto, aby byly “zalohovany” a lepe si mohl delit zatez
      * kazdy prostor oddelen teckami
      * je bohaty a stale se obohacuje
        + zakladni domeny: edu, com, gov, mil, met, org, …
        + narodni domeny: cz, sk, ca, us, jp, fr, … (prislusnost uzlu ke statu)
        + dalsi: eu, net
    - HTTP (hyper-text transfer protocol)
      * vyuzivany na www
      * prenaseny obsah v ruznych formatech
      * zakladni model client/server
      * pouziva se TCP
      * hypertext
        + v textu kodovane odkazy, ktere ukazuji na dalsi stranky
        + jeden z mala evropskych vynalezu (vznikl v CERNu)
      * dotazujeme se pomoci URL (uniform resource locator) - definice zdroje, ktery chce klient dostat
        + soucasti: protokol, uzel, port, cesta k informaci
      * kategorie dokumentu
        + staticke - s pevnym obsahem
        + dynamicke - neexistuji v predem stanovenem formatu a jsou tvoreny podle pozadavku klienta
        + aktivni - serverem je poskytnut program, ktery je spousten u klienta (napr. java applety)
    - FTP (file transfer protocol)
      * velmi stary protokol
      * standardni mechanismus urceny pro prenos souboru mezi uzly
      * model client/server
      * pouziva 2 TCP spojeni
        + spojeni prenasejici ridici zpravy (std port 21) - udrzovano po celou dobu realce
        + spojeni prenasejici data (std port 20) - otevira se znovu pro kazdy prenaseny soubor
      * ridici komunikace
        + domluva parametru spojeni - jaky typ souboru prenaset, jaka je vnitrni struktura souboru, jaky bude prenosovy mod (proudovy/blokovy/komprimovany) → nutnost pro preklenuti heterogennosti
      * datova komunikace
        + prenaseji se vlastni data
* multimedia v sitich
  + cil: prenos zvuku a obrazu po pocitacove siti
  + rostou pozadavky na kvalitu spojeni
    - potreba vcasneho doruceni
    - neprilis velky rozptyl doruceni paketu
  + spojovane site (telefony)
    - jednodussi reseni
    - nedostatecna koncova kapacita
    - potencialni plytvani pasmem (musi byt vyhrazeno, i kdyz mlcite)
  + prepinane site
    - mohou dobre vyuzit multicast
    - vyzaduji kvalitu sluzby: rezervace
    - mozna reseni
      * overprovision (dostatek kapacity bez ohledu na pozadavky)
        + mame obrovsky nadbytek zdroju (napr. paterni site) → neni potreba zadne rezervace
      * dedikovane okruhy (a la telefony): VPN
        + okruh je trvale nas bez ohledu na to jestli ji pouzivame nebo ne
      * rezervace pro kazdy tok zvlast: RSVP
        + dela rezervaci az kdyz chceme komunikovat
        + postupne se zabira kapacita na jednotlivych castech a overujeme, ze mezi mnou a cilem je dostatecna kapacita
        + informuje aktivni prvky o tom co si zabirame - tzn. vzdy smerovac vi kolik jeste muze poskytnout
        + je to dost narocne na vykon aktivnich prvku
        + moc se nepouziva - pouze ve specializovanych a mensich sitich
      * agregace toku, rezervace (staticka) pro agregace: DiffServ
        + rekneme, ze jista skupina uzivatelu muze komunikovat takovym kanalem
        + kazdy uzivatel je nejak privilegovan
        + je dobre skalovatelny
        + vhodne reseni pro soucasny internet
  + streaming
    - zpusob doruceni multimedialniho obsahu klientum
    - neni interaktivni (prijimajici strana je pasivni)
    - live streaming
      * multimedialni obsah vznika zive behem streamovani
    - video on demand vs. pasivni prijem
      * pasivni prijen obvykle pro prijem zivych streamu
      * mozne streamovat i multimedialni archivy
    - muze byt velke mnozstvi odberatelu
  + videokonference
    - jednoznacny pozadavek na interaktivitu
      * data se posilaji obousmerne
    - obvykle malo odberatelu
* bezdratove site
  + cil: umoznit pristup k vypocetnim a komunikacnim zdrojum mobilnich zarizeni
  + s infrastrukturou: bunkova sit
  + bez infrastruktury: ad-hoc site
    - zarizeni-zarizeni
  + charakteristiky
    - podstatne vyssi chybovost
      * oprava problemu musi byt primo na spojeni, ne na vyssich vrstvach
        + casto formou samoopravnych kodu
      * je to narocnejsi na kontrolu nez napr. u optiky
      * casto kombinovano s redundanci
    - opticke site (infra, laser)
      * jsou silne zavisle na vnejsich podminkach (mlha)
    - radiove
      * kapacita zavisla na frekvenci, kvalita na kodovani a vyzarene energii
      * vice standardu
  + bunkova sit
    - zakladove stanice pokryvaji uzemi signalem
    - zakladny jsou propojene dratovou siti
    - veskera komunikace mobilnich agentu je smerovana pres zakladove stanice
    - mobilni agent muze plynule prechazet mezi zakladovymi stanicemi



* + - lze kontrolovat kdo je pripojen, lze kontrolovat prenos, lze uctovat
  + ad-hoc site
    - modivace: vytvorit sit pri absenci infrastruktury
      * zivelne katastrofy, nedostatek financi/casu
      * vyuziva pouze sitove vlastnosti ucastniku
    - princip
      * kolekce autonomnich uzlu komunikujicich skrze decentralizovanou milti-hot sit
      * kazdy uzel zaroven koncovym uzlem i sitovym smerovacem
      * dynamicka topologie site
      * rizeni site rozdeleno mezi jednotlive uzly
    - vyhody
      * rychle budovani
        + nikdo nemusi tvorit pristupove body → staci nakonfigurovat jednotlive ucastniky
      * odolnost - neobsahuji single point of failure
        + site dynamicky reaguji na zmeny v siti bez problemu a padu
      * efektivni vyuzivani radioveho spektra
    - nevyhody
      * omezeny dosah bezdratove komunikace
      * komplikovane rizeni site diky neexistenci centralni entity
      * zmeny v topologii pri pohybu mobilnich uzlu
        + velmi slozite algoritmy
    - aplikace
      * zachranne operace pri prirodnich katastrofach
      * zasitovani osobnich zarizeni (hodinky, PDA, medicinske pristroje)
      * vojenske operace
      * senzorove site
  + mobilni pocitani
    - co chceme pocitat (delat) v bezdratove siti
    - mozne realizace
      * always on - bezdratove site
      * prenos prostredi - realizovane softwarove
    - mobilita s prenosem prostredi
      * napr. cteni posty pres webovy prohlizec
      * problemy
        + ruznost klientskych systemu
        + bezpecnost - autentizace uzivatele
        + vnimana kvalita zavisla na kvalite pripojeni
    - snaha o unifikaci a jednoduchy prenos prostredi (napr. cloudove sluzby)
      * drive to havarovalo nejcasteji na vykonu systemu a site → bylo to moc pomale a nedalo se to pouzivat
* distribuovane systemy
  + definice:
    - system, ktery je tvoren dvema nebo vice nezavislymi pocitaci propojenymi siti a komunikujicimi formou predavani zprav
    - distribuovany system tvori nezavisle pocitace, ktere se uzivateli jevi jako jeden celek
  + propojeni vice pocitacu pomoci site do “jednoho”
  + napr.
    - internet - ale neni to uplne pravda, ale mame pristup obvykle pres jeden system → internet se jevi jak “jednotka”
    - telefonni system (automaticke ustredny)
    - multimedialni systemy (videokonference, e-learning)
      * spojeni n-ucastniku - vidi se navzajem, ale nevidi co stoji “mezi nimi” a umoznuje jim komunikaci
    - mobilni systemy
    - clustery
      * vice pocitacu sdilejici vykon (spolecne resi nejakou ulohu)
    - gridy
      * distribuovane vypocetni systemy → v ruznych koncinach sveta
    - peer-to-peer systemy
      * napr. sdileni dat a obsahu
      * uzivatel → uzivatel (zadna infrastruktura - pouze uzly)
    - cloud
      * prostredi ze ktereho mohu ziskat dalsi vykon, sdileni prostredku
  + problemy
    - heterogenita jednotlivych slozek
      * ruzne komponenty v systemu → je nutne, aby se byly schopny domluvit
      * middleware: skryva heterogenitu (CORBA, Globus) → zamaskuje se
      * mobilni kod (java - lze ji pustit na ruznych platformach)
        + vyuziti virtualni pocitacu (plna, JVM, …)
    - otevrenost/interoperabilita
      * nezbytne vyuziti standardu → jinak nemuze byt celek propojeny
    - bezpecnost
      * autentizace, autorizace, soukromi
      * moznost anonymity
    - zpracovani vypadku
      * detekce, maskovani, tolerance
    - rozsiritelnost
    - paralelismus
      * nebezpeci napr. “smrtelneho objeti” (deadlock)
        + dva stroje cekaji sami na sebe
      * zavislosti (synchronni pohled)
    - transparence (nestaram se co kde je)
      * pristup
      * misto
      * replikace
      * selhani
      * mobilita/prenositelnost
      * vykon
      * skalovatelnost/rozsiritelnost
* cloud computing
  + novy pristup k nabidce vypocetnich a uloznych sluzeb
  + postaven na virtualizaci zdroju
    - ta umoznuje nabidnout pocitac nebo celou skupinu pocitacu pri zachovani spravovatelnosti
    - uzivatel dostava “holy” system, ktery sam spravuje
    - jednoduchy pristup - zpravidla pres webove rozhrani
    - pay per use (tj. zadne pocatecni investicni naklady)
  + napr. Amazon Elastic Cloud
    - pristup pres www rozhrani
    - platba kreditni kartou
  + cloudy nabizi flexibilni kapacitu
    - je mozne okamzite dokoupit dalsi zdroje
    - virtualizace podporuje navysovani vykonu poskytnutim kiopii
    - potencial pro odolnost proti vypadku
  + otevrene bezpecnostni problemy
    - nevime, kde jsou data
    - data i vypocty jsou outsourcovany
* peer-to-peer systemy
  + decentralizovany distribuovany system; klient-klient
  + tvoren vzajemne komunikujicimi identickymi entitami (peery)
  + opak modelu client-server
  + kazdy peer je zaroven serverem i klientem
    - poskytuje sluzby ostatnim peerum → role serveru
    - vyuziva sluzby ostatnich peeru → role klienta
  + napr.
    - skype, bioinc, seti@home, bittorrent, bitcoin
  + vlastnosti
    - distribuovane rizeni - neexistuje centralni entita
      * pro vstup staci znat adresu jen jednoho peera
      * nehrozi pretizeni centralni entity
    - samoorganizace
    - heterogenita - peerove bezi na ruznych platformach
    - skalovatelnost - nehrozi pretizeni centralni entity
    - dynamika - topologie systemu se velmi meni
      * uz z principu je
    - sdileni zdroju - kazdy peer se svymi zdroji podili na fungovani P2P systemu
  + srovnani s klient-server
    - narocnost zbudovani
      * K-S vyuziva jednoduchych modelu komunikace
      * P2P vyzaduje komplexni interakce
    - spravovatelnost
      * sprava KS systemu je prehlednejsi diky koncentraci komunikace v jednom bode
      * u P2P se slozite udrzuje velke mnozstvi zdroju
    - skalovatelnost
      * KS je limitovan HW parametry centralniho uzlu → ale vyuziva se vyvazovani zateze mezi nekolika fyzickymi stroji
      * P2P system skaluje z principu - s rostoucim poctem peeru roste kapacita systemu
    - bezpecnost
      * v KS modelu je za bezpecnost zodpovedny server
      * v P2P je zodpovednost rozlozena mezi peery - nutnost komplexnejsich bezpecnostnich protokolu
    - spolehlivost
      * KS je zavisly na behu serveru → single point of failure
      * P2P system je do velke miry redundantni - jednu funkcionalitu poskytuje zaroven vice peeru
  + ma velky smysl do budoucnosti
  + architektura
    - strukturovane
      * vyuzivaji samoorganizacnich algoritmu
      * vytvareji predem danou topologii
      * distributed hash table
        + jsou vyuzivany k tomu, aby se uzel rozhodl kam posle dotaz
        + podobne jako na treti vrstve v ISO/OSI modelu
      * narocne pripojeni dalsiho uzlu - musime ho zaclenit do vytvarene struktury
      * ale zase se jednoduse hledaji informace v siti
    - nestrukturovane
      * peerove se pripojuji ad-hoc
      * nesnazi se dat propojenim zadnou pevnou strukturu
      * jednoduche pripojeni dalsich klientu
      * ale problem: jak najit danou informaci?
        + nahodna prochazka
        + zaplava
        + pomale a neefektivni (nekdy informaci v rozumnem case nejsme schopni nalezt)
    - centralizovane
      * centralni server, ktery poskytuje pouze indexovaci sluzbu
      * zjednoduseni spravy a distribuovani poskytovanych informaci
        + ale problem je s rostoucim poctem uzivatelu je hlavni server cim dal tim vic zahlcovany
    - hybridni
      * nektere uzlu povyseny na superuzly
      * organizace site a komunikace mezi komponentami prenesena na superuzly
* senzorove site
  + autonomni senzory schopne vzajemne komunikace
  + cil: merit fyzikalni a chemicke veliciny na rozsahlem uzemi
  + pozadavek na skalovatelnost na velka uzemi vylucuje “dratove” site → vyuziti bezdratove komunikace
  + casto spojovane s ovladacimi prvky
  + velkym problemem je malo elektricke energie → nutnost efektivni komunikace a algoritmu
  + napr.
    - monitoring znecisteni ovzdusi
    - sledovani sireni lesnich pozaru
    - detekce naruseni hlidaneho prostoru
    - inteligentni budovy, monitoring struktury staveb
    - sledovani stavu zemedelskych ploch
  + senzory
    - velmi male a jednoduche
    - napajeni z baterie nebo ziskani energie z prostredi
      * napr. miniaturni solarni panel
    - velke mnozstvi, pokryvaji potencialne rozlehla uzemi
    - schopne komunikace mezi sebou
    - prevazne datove trendy s nizkou frekvenci
      * snaha vymyslet protokoly druhe a treti vrstvy, aby senzory sit vytvorily a data pres sebe poslaly na base station
  + sit
    - decentralizovana
      * nema zadnou infrastrukturu (prileti letadlo a rozhodi senzory nad pozadovanym uzemim)
    - samoorganizujici mechanismy pro spravu
    - znacne homogenni, maly pocet hranicnich (sbernych) uzlu
      * senzory obvykle stejneho typu
  + problemy
    - rozmisteni senzoru: (ne)deterministicke, (semi)autonomni
      * kolik jich rozhazet po okoli?
    - smerovani dat
    - sber a distribuce dat z/do site
    - fuze senzoru: robustnost skrze redundanci
    - energeticka narocnost algoritmu
      * viz. vyse
      * nejkratsi cesta vs. energeticky nejlevnejsi cesta
      * zatez je treba rozkladat prubezne mezi jednotlive uzly
    - nespolehlivost senzoru i komunikace
      * poskozeni, ztrata napajeni senzoru nesmi vyradit celou sit
      * selhavajici uzel se muze chovat jako zaskodnik
      * fuzovane uzly se musi dohodnout na spolecne hodnote
* socialni site
  + samotne site jsou starsi nez IT → moderni technologie jsou jen prostredek
  + sociologicke pojeti
    - skupiny definovane vzajemnymi propojujicimi vztahy
    - pratelstvi, obchodni styk, pribuzenske vztahy, spolecne zajmy
    - znazornovana jako graf
      * je to nejvhodnejsi znazorneni
  + socialni sitovani jako sluzba
    - webova aplikace reflektujici socialni site z realneho sveta
    - umoznuje aktivni vytvareni novych siti
  + analyza grafu
    - vzdalenost dvou uzlu - teorie sesti stupnu odlouceni
      * existuje mezi 2 libovolnymi lidmi na zemekouli vzdalenost maximalne 6 “hopu” (hran v grafu)
    - rozklad na komponenty - jednotlive socialni skupiny
    - studium sireni informaci - epidemiologicke metody
    - egocentricka analyza - socialni kapital jedince
  + historie
    - prvni snahy uz na pocatku internetu
    - 90. leta - zajmove komunity
    - koncem tisicileti nova generace: sixdegrees, friendster, myspace, linkedin, facebook
  + charakteristika
    - uzivatele vytvareji a udrzuji svuj profil (soubor osobnich informaci)
    - uzivatele definuji uni/bilateralni vztahy
    - uzivatel muze prohlizet a prochazet svuj seznam kontaktu a do jiste miry i propojeni mezi ostatnimi
  + niche communities
    - nektere socialni skupiny jsou i v realne svete uzavrene
    - reflektovano v sitovych sluzba: uzavrene skupiny, definovani okruhu kontaktu, …
  + bezpecnost
    - kontrola toku osobnich informaci
    - rozdeleni dat na soukroma a verejna
    - menici se pravidla stezuji kontrolu
* multimedia
  + typy
    - zvuk
    - obraz
    - haptika - dotek
  + zpracovani
    - akvizice
      * jak je dostat?
    - uchovavani
      * uchovat, poslat siti
    - prezentace
      * prevod z PC formatu pro lidi
  + nedokonale vnimame
    - napr. zpozdeni, cast chybejicich dat
    - tim padem toho lze dost dobre vyuzit
  + zvuk
    - podelne mechanicke vlneni v latkovem prostredi, ktere je schopno vyvolat sluchovy vjem
    - podleha beznym fyzikalnim jevum (odraz, ohyb, difrakce)
    - vjeho vnimani je slozity proces - jeste nemame presne zdokumentovane jak to funguje → jsou jen teorie, ktere pouzivame
    - frekvencni rozsah: 20 Hz az 22 kHz
    - s rostoucim vekem horni hranice vyrazne klesa
    - dynamicky rozsah
      * rozdil mezi nejhlasitejsim a nejtissim vnimatelnym zvukem
      * uprostred slysitelneho frekvencniho pasma asi 120 dB
      * na okrajich pasma je mnohem mensi
    - rozlisovani frekvence
      * schopnost rozlisit frekvence tonu
      * u kazdeho cloveka jina a frekvencne zavisla
      * uprostred slysitelneho frekvencniho pasma vyssi nez na okrajich
    - frekvencni maskovani
      * schopnost odlisit dva frekvencne blizke tony
      * jeden ton muze potlacit slysitelnost jineho, soucasne znejiciho tonu
      * maskovaci schopnosti zavisle na
        + frekvenci maskujiciho tonu
        + frekvencni vzdalenosti obout onu
        + sirkou kritickeho pasma
      * vyuzivane v nekterych kompresnich algoritmech (MP3)
    - casove maskovani
      * potlaceni slysitelnosti mene hlasiteho tonu po hlasitem
      * potlaceni ticheho tonu predchazejiciho maskovacimu tonu
    - zpracovani
      * akvizice zvuku
      * diskretni reprezentace spojiteho zvukoveho vlneni
      * zpracovani diskretnich zvukovych dat
        + analyza zvuku za pouziti psychoakustickych modelu
        + pouziti zvukovych filtru: ekvalizace, odstraneni sumu a echa
        + datova komprese
      * ulozeni na datovy nosic
      * prenos po siti
    - A/D prevodnik
      * typicky zvukova karta v PC, ale i jina zarizeni
      * prevod analogoveho zvuku do digitalni podoby a naopak
    - vzorkovani
      * odebirani vzorku signalu v definovanych casovych intervalech (vzorkovaci frekvence)
      * prevadi spojity casovy prubeh signalu na diskretni reprezentaci
    - kvantovani
      * diskretni reprezentace hodnoty intenzity zvuku
      * 1bit odpovida priblizne 6 dB
    - vzorkovaci frekvenci v praxe:
      * 8 kHz, 8 bitu na vzorek - telefony
      * 11 kHz - lidska rec
      * 22 kHz - audiokazety
      * 44,1 kHz - CD
      * 48 kHz, 96 kHz, 192 kHz, 24 bitu na vzorek - DVD
      * 64 bitu na vzorek - sonary
    - PCM (pulse code modulation)
      * analogovy signal → vzorkovani → kvantovani → PCM
      * datove toky
        + 8 kHz x 8 b x 1 kanal = 64 kbps
        + 192 kHz x 24 b x 2 kanaly = 9000 kbps
      * pouziti: telefony, CD, DVD
      * bezne implementovane v HW
    - CD
      * red book audio standard (1980)
      * 2 audio kanaly
      * vzorkovaci frekvenci 44,1 kHz, 16 bitu na vzorek, PCM
      * detekce a oprava chyb
      * ochrana proti kopirovani porusuje red book standard
    - DVD
      * vysoce verny (hi-fi) format
      * mnoho moznych konfiguraci kanalu (1.0 az 7.1)
      * oproti CD: vetsi vzorkovaci frekvence, vice bitu na vzorek
      * maximalni mozny bitrate pro vsechny kanaly je 9,6 Mbps
    - komprese
      * MPEG-I Layer III
        + ztratory kompresni mechanismus
        + standardizovany v roce 1991
        + komprimuje PCM audio v pomeru priblizne 1:11
        + pouziti psychoakustickeho modelu

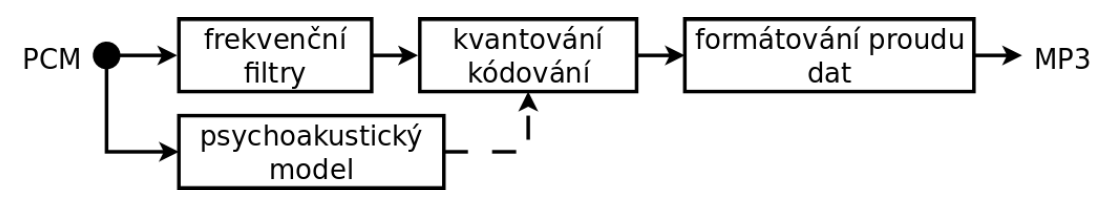
zahazuje casti signalu hure rozeznatelne pro lidske ucho

podle charakteru signalu vybira nejpodstatnejsi frekvencni pasma

* + - * + komprese signalu

komprese probiha ve frekvencni domene

nelinearni kvantovani



* + - * MPEG-I Layer II
        + DVD, DVB-T, interne v radiich
        + sub-band kodovani audia

v casove domene

signal se rozdeli na 32 frekvencnich pasem, na zaklade psychoakustickeho modelu se nektera zahodi a ostatni se komprimuji samostatne

* + - * Vorbis
        + open-source obdoba MP3
        + pokrocilejsi kvantovani, redukce sumu na zaklade statistickych mereni
      * AAC
        + technologicky nastupce MP3
        + silne vyuziti psychoakustickych modelu
        + samoopravne kody
        + modularni kodovani

paleta profilu pro ruzne tridy signalu

umoznuje dosahnout lepsich vysledku pri soucasnem snizeni bitrate

* + - * + az 48 kanalu, temer libovolne vzorkovaci frekvence
      * AC3
        + aka Dolby Digital
        + az 8 kanalu, pevny bitrate
        + srovnatelny s MP3
      * Windows media audio (wma)
        + podpora DRM
        + paleta kodeku pro ruzne ucely (hlas, ztratove/bezztratove)
        + kvalita srovnatelna s MP3
      * Speex
        + opensource ztratovy format urceny pro kompresi reci
        + pro samohlasky vyssi bitrate, pro sykavky nizky
        + vzorkovaci frekvence 8kHz, bitrate 2 kbps az 44 kbps
        + robustni proti vypadkum behem prenosu
      * realauido
        + proprietarni format vhodny pro streaming
        + bitrate 6 az 176 kbps, optimalizace na rec, hudbu, dolby surround
        + obalka pro celou radu kodeku
      * FLAC
        + opensource bezztratovy format
        + kodovani s celociselnou presnosti eliminuje chyby pri zaokrouhlovani
        + libovolna vzorkovaci frekvence
        + kvantovani vzorku 4-32 bity
        + kompresni pomer 20-30% ve srovnani s PCM
  + obraz
    - svetlo
      * elektromagneticke vlneni s velmi uzkou sirkou spektra (viditelne svetlo) odrazene od objektu v okoli a dopadajici na svetlocitlive bunky sitnice oka
    - vnimani obrazu
      * obrazova pamet
        + clovek nevnima obraz kontinualne, ale v “kvantech”
        + obrazovy vjem jsme schopni zpracovat priblizne kazdych 60 ms → jsme schopni spojit posloupnost obrazku do pohybu
        + oko ma solidni toleranci chyb (az vypadky do 5% nejsme schopni zaznamenat)
      * oko nevnima vsechny slozky obrazu stejne (nejcitliveji jas)
      * clovek je vice senzitivni k chybam v audiu nez ve videu
      * oko nevidi prirozene zcela ostre
      * mozek ma tendence si domyslet casti vjemu na zaklade zkusenosti
      * az 80% naseho mozku zpracovava co vidime → velmi narocne
    - formaty
      * vzorkovani: rozdeleni obrazu pri akvizici na jednotlive body (vzorky)
      * SDTV formaty
        + rodina standardu popisujici rozliseni a framerate videa
        + PAL: 720 x 576 obrazovych bodu, 25 fps
        + NTSC: 720 x 480 obrazovych bodu, 29.97 fps
      * HDTV formaty
        + udava se v poctu radu v rozliseni
        + progresive frames (p) nebo interlaced (prokladane) fields (i)
        + 1080i: 1920 x 1080 (30i, 29.97i)
        + 720p: 1280 x 720 (60p, 59.94p, 30p, 29.97p, 24p, 23.97p)
    - framerate
      * pocet obrazovych snimku za sekundu
      * nejstarsi technologie zacinaly na 6 - 8 snimcich za sekundu
      * v soucasnosti jsou nejpouzivanejsi 25fps, 29.97 fps, 24 fps
      * lze se setkat i s jinymi hodnotami
      * pro dosazeni iluze pohybu je treba 10 fps
        + ale zalezi na scene - napr. pro pohled krajinou to staci, pro akcni scenu uz spise ne
      * psychologicka hranice skutecne pouzitelne iluze pohybu je spise 12.5 fps
      * vetsi pocet snimku za sekundu pomaha skryvat nedokonalosti komprese
    - akvizice
      * videokamery
        + analogove (ctvercove pixely), digitalni (DV) - PAL, NTSC
        + HDTV
      * analogove grabovaci karty
        + podpora formatu PAL/NTSC
        + HW podpora kodovani videa (MPEG2)
      * akvizice
        + obrovske datove toky (1.5 Gbps pro 1080i HDTV video)
    - komprese
      * MPEG
        + 3 typy snimku

I frame - Intrapicture, referencni snimek

P frame - predicted, rozdil oproti predchozimu snimku

B frame - bidirectional predicted, interpolace mezi predchozim a nasledujicim snimkem

* + - * + rozklad obrazu do makrobloku
        + B frames - motion vector zmena v ramci makrobloku oproti predchozimu snimku
      * MPEG-2
        + podporuje kodovani interlaced (prokladaneho) videa
        + MPEG-2 kodovani neni optimalizovane pro bitrate nizsi nez 1 Mbps
        + nekolik profilu rozdelenych do urovni podle komplexity komprese
        + typicke pouziti: DVD, DVB, HDV, HDTV
      * MPEG-4
        + cela rodina ruznych standardu
        + definuje obalkove formaty, audio a video formaty, titulky, interaktivni prvky (DVD menu)
      * DV
        + neni zatizen licencnimi poplatkami
        + pouziti ve spotrebni elektronice (digitalni kamery, nektere DVD rekordery, …)
        + pevny datovy tok 25 Mbps
      * Theora
        + opensource navrh i implementace
        + prima konkurence pro MPEG-4 kompresni mechanismy
        + obsahuje pouze I a P frames
    - obalkove formaty
      * obaluji vlastni data a uchovavaji pripadna metadata
      * AVI (audio video interleave)
        + obalkovy format MS z roku 1992
        + nepodporuje titulky a moderni kodeky
      * MOV
        + obalkovy format pro QuickTime
        + jedna nebo vice stop pro audio, video, efekty nebo text
      * Matroska
        + opensource navrh obalkoveho formatu
        + vystaveny na binarnim ekvivalentu XML (EBML)
        + rychle prochazeni audiem nebo videem (seek)
        + implementuje menu jako u DVD, podpora pro streaming