Graf G je usporiadana dvojica mnozin (V, E), kde V je mnozina vrcholov a E je mnozina hran – mnozina  
dvojprvkovych podmnozin mnoziny V.

Vrcholy spojene hranou nazyvame susedne, hrana je incidentna k vrcholom, ktore spaja.

Orientovany graf G je graf, ktoreho hrany su usporiadane dvojice vrcholov (u,v), hrana potom vychadza z vrcholu u a vstupuje do vrcholu v.

Multigraf je graf, ktory nahradza mnozinu hran multimnozinou a umoznuje existenciu sluciek – hran spajajucich vrchol sam so sebou.

Ohodnotenie grafu – priradenie prvku konecnej mnoziny vrcholom a hranam grafu

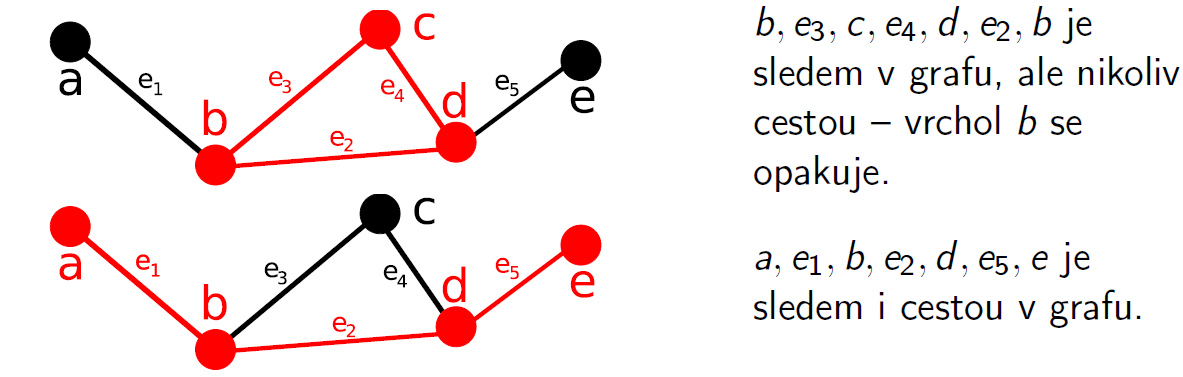
Priklady ohodnotenia na sietach:  
Ohodnotenie vrcholov: ohodnotenie sietovych zariadeni L2 a L3 adresami  
Ohodnotenie hran: Sirka pasma, latency, cena prenosu  
Ohodnotenie vrcholov nazvom stavov, hran typom sprav pri abstraktnom navrhu protokolov

Stupen vrcholu v neorientovanom grafe je pocet hran incidentnych k vrcholu.

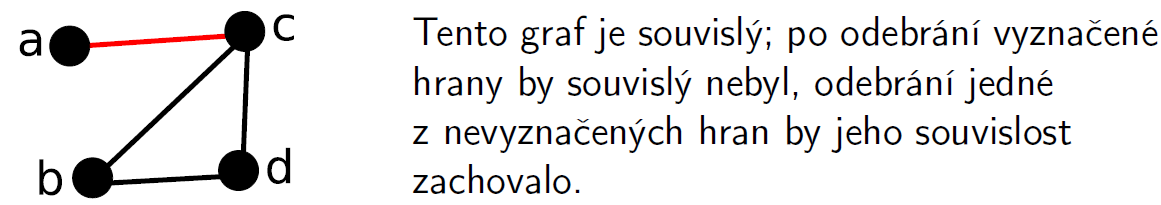
V orientovanom grafe sa rozlisuje vstupna a vystupna hrana. Vstupnym stupnom vrcholu orientovaneho grafu je pocet vstupujucich hran do vrcholu, vystupnym stupnom vrcholu je pocet vystupujucich hran z vrcholu, celkovy stupen vrcholu je sucet vstupneho a vystupneho stupna vrcholu.

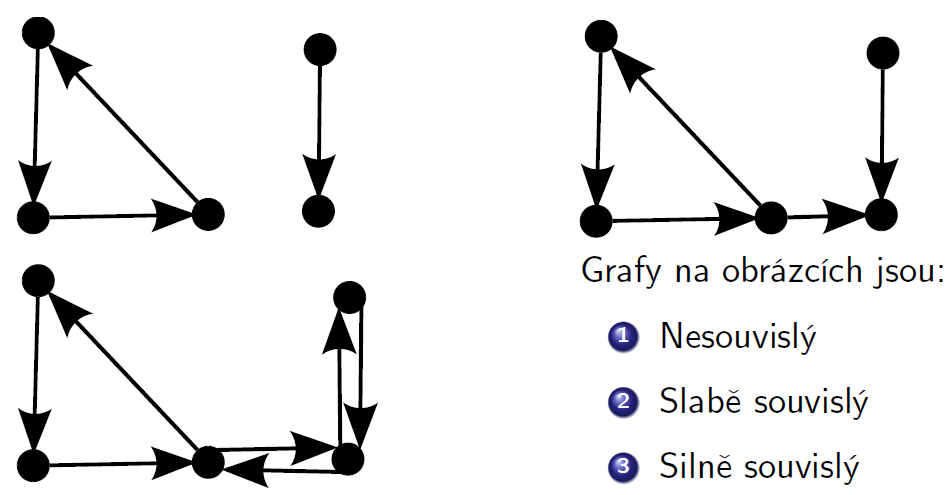
Sledom v grafe nazyvame postupnost vrcholov a hran v0,e1,v1,…,en,vn, kde kazda hrana ei spaja vrcholy vi-1,vi, resp. vedie z vi-1 do vi. Samostatny vrchol je tiez sledom.

Cesta v grafe je sled bez opakovania vrcholov.



Graf je suvisly, ak medzi jeho kazdymi dvoma vrcholmi vedie cesta.  
Ak nahradime vsetky hrany orientovaneho grafu G neorientovanymi a ziskame suvisly graf => graf G je slabo suvisly.  
Orientovany graf je silno suvisly, ak medzi jeho kazdymi dvoma vrcholmi vedu cesty v oboch smeroch.





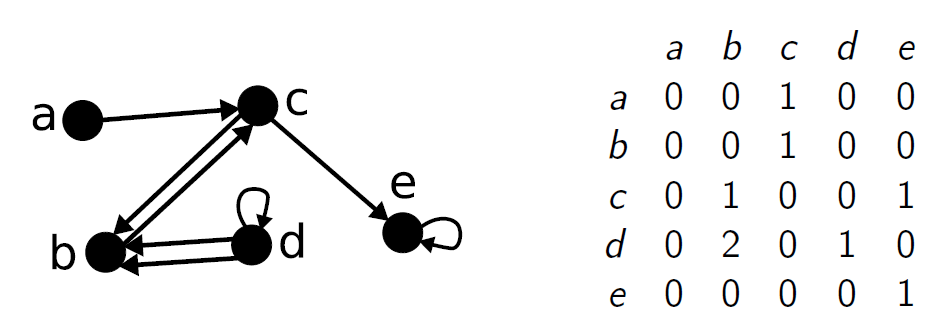
Ukladanie grafu do pocitaca alebo aktivneho sietoveho prvku 2 sposobmi:  
1. Matica susednosti  
2. Zoznamy susedov

**MATICA SUSEDNOSTI**

matica E o rozmeroch n x n pre n vrcholov grafu  
Eij = 1, ak hrana (i,j) patri do grafu, Eij = 0, ak hrana (i,j) nepatri do grafu  
pre neorientovany graf je symetricka, pre orientovany nemusi byt symetricka

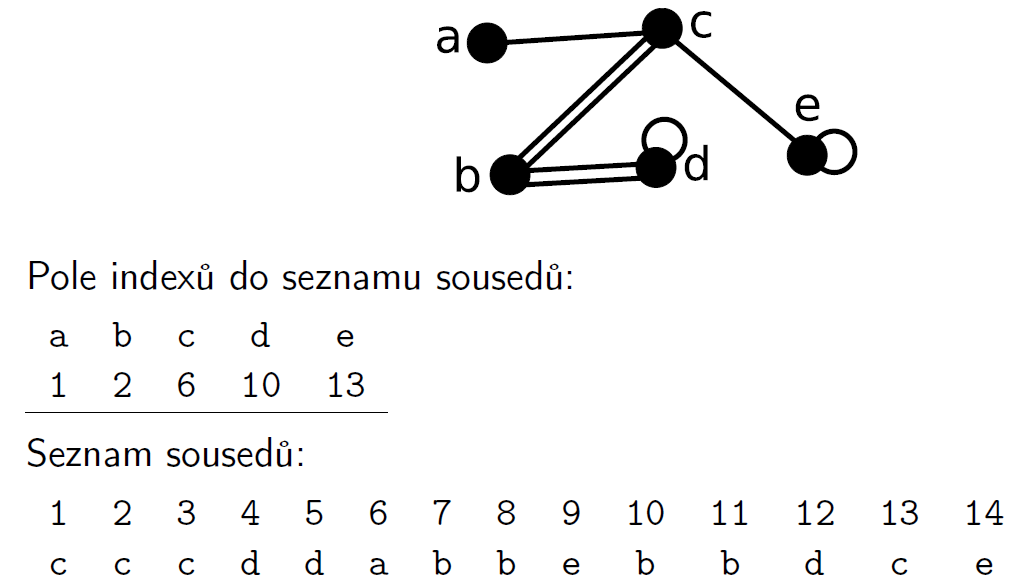






**ZOZNAMY SUSEDOV**

pre kazdy vrchol existuje samostatny zoznam susedov, s ktorymi je vrchol spojeny hranou (alebo do nich vedie orientovana hrana)  
Implementacia pomocou dvoch jednorozmernych poli:  
 -v jednom su ulozene vsetky zoznamy za sebou, zoradene podla cisla vrcholu  
 -druhe uchovava indexy, na ktorych zacinaju v prvom poli susedia kazdeho vrcholu  
Nasobne hrany v multigrafe su zadane nasobnym uvedenim vrcholu v zozname susedov

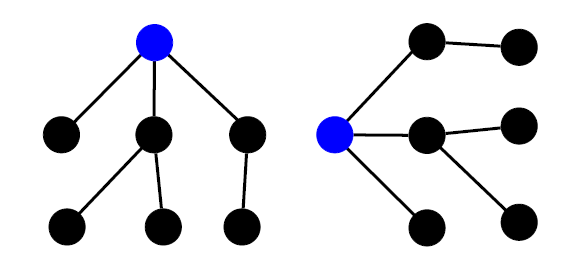


Trivialna cesta obsahuje jeden vrchol.

Kruznica v grafe je netrivialna neorientovana cesta, ktora zacina aj konci v rovnakom vrchole.   
Orientovana kruznica je kruznica zlozena z orientovanych hran, respektujucich orientaciu tychto hran.

Les je graf, ktory neobsahuje kruznice.  
Strom je graf, ktory neobsahuje kruznice a je suvisly.

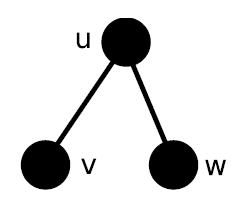
Korenovy strom – orientovany strom, ktory ma urceny koren a z tohto korena vedie orientovana cesta do vsetkych ostatnych vrcholov, vyuzitie napr v DNS, Multicast

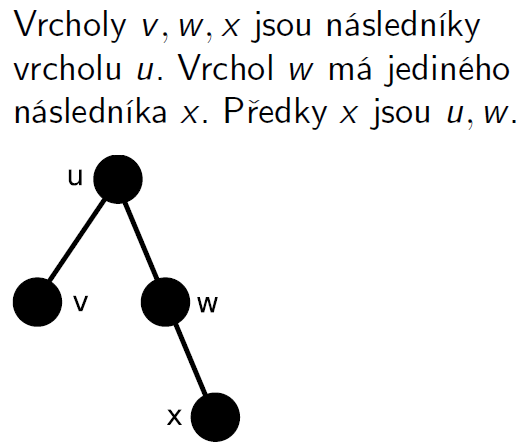


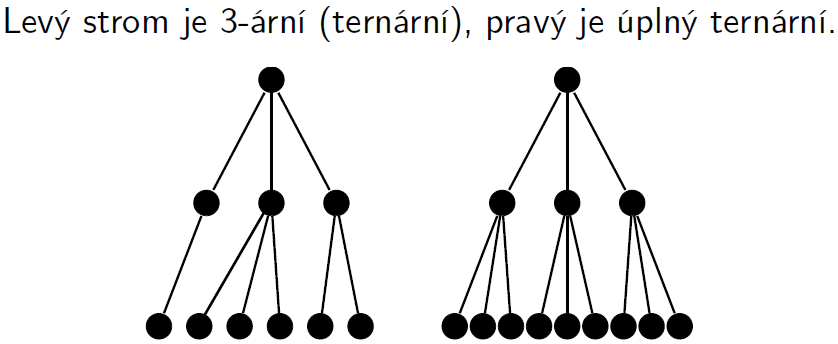
Orientovany strom je korenovy prave vtedy, ked prave jeden z jeho vrcholov ma vstupny stupen 0 a vsetky ostatne vrcholy 1.

Hlbka/uroven vrcholu – vzdialenost (pocet hran na ceste) vrcholu od korena stromu.  
Vyska korenoveho stromu – najvyssia z hlbok vsetkych jeho vrcholov.

Ak vedie v korenovom strome hrana z vrcholu u do vrcholu v => u je rodicom v a v je synom u, ak maju vrcholy spolocneho rodica, nazyvame ich surodencami.



Vrchol u je predkom vrcholu v, ak lezi na ceste z r do v, v sa v takom pripade nazyva naslednikom u.  
Vrchol, ktory nema ziadnych potomkov, nazyvame list stromu.  
Ostatne vrcholy sa oznacuju ako vnutorne.

N-arny strom – korenovy strom, ktoreho kazdy vrchol ma nanajvys n potomkov.  
Uplny n-arny strom – korenovy strom, ktoreho vnutorne vrcholy maju prave n potomkov a vsetky listy maju rovnaku hlbku.

Usporiadany strom – korenovy strom s danym poradim potomkov kazdeho vrcholu.

Pre kazdu uroven n-arneho stromu, je dany limit poctu vrcholov v tejto urovni.  
V m-tej urovni n-arneho stromu, sa nachadza nanajvys nm vrcholov.

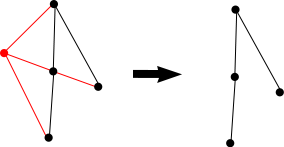
Kazdy korenovy strom sa da previest na binarny.  
Vnutorne algoritmy smerovacich zariadeni, mozu byt zalozene na binarnych stromoch.

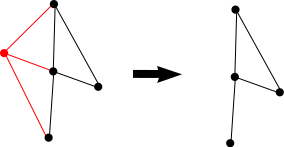
Graf H=(VH,EH) je podgraf grafu G=(VG,EG), ak su splnene nasledujuce podmienky:  
 1. VH ⊆ VG  
 2. EH ⊆ EG  
 3. Hrany grafu H maju oba vrcholy v H

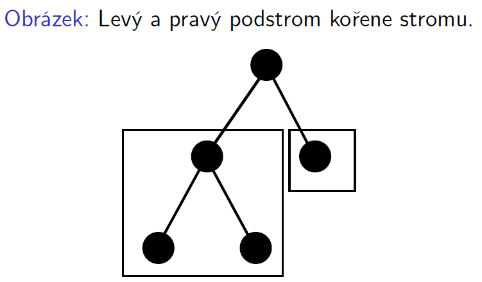
* Podgraf vznikne vymazanim niektorych vrcholov povodneho grafu, vymazanim vsetkych hran   
  povodneho grafu, ktore nemaju oba vrcholy v podgrafe H, tj. vymazanim hran, ktore zasahuju do vymazanych vrcholov a pripadnym vymazanim dalsich hran.

Graf H=(VH,EH) je indukovany podgraf grafu G=(VG,EG), ak je podgrafom grafu G a pre kazde dva vrcholy   
u,v podgrafu H plati: (u,v) ∊ EG -> (u,v) ∊ EH

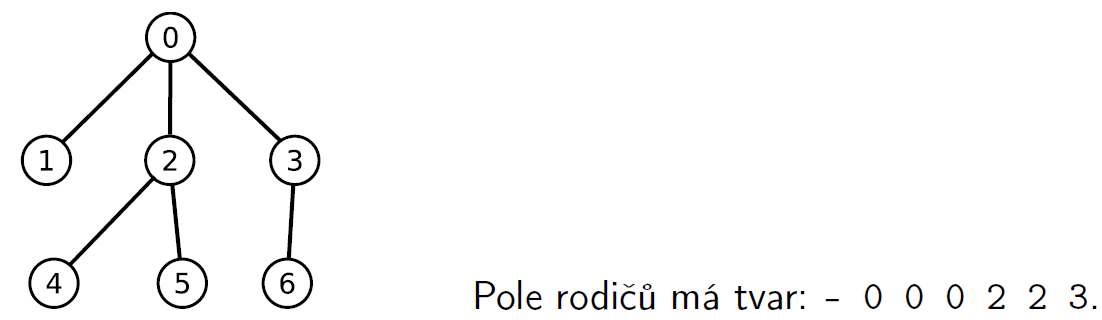
* Indukovany podgraf vznikne vymazanim niektorych vrcholov povodneho grafu, a vymazanim  
  len tych hran povodneho grafu, ktore nemaju oba vrcholy v podgrafe H, tj. hran, ktore zasahuju  
  do vymazanych vrcholov.

Indukovany podgraf binarneho stromu G, tvoreny jednym potomkom vrcholu v a jeho nasledovnikmi, sa nazyva podstromom vrcholu v a stromu G.  
Kazdy vrchol ma lavy a pravy podstrom, pricom jeho lavy, resp. pravy potomok je korenom tohoto podstromu.   
Pravy a lavy podstrom binarneho stromu o vyske h maju vysku nanajvys h-1, pricom najmenej jeden z nich ma vysku prave h-1.

 *Povodny graf a jeho podgraf*

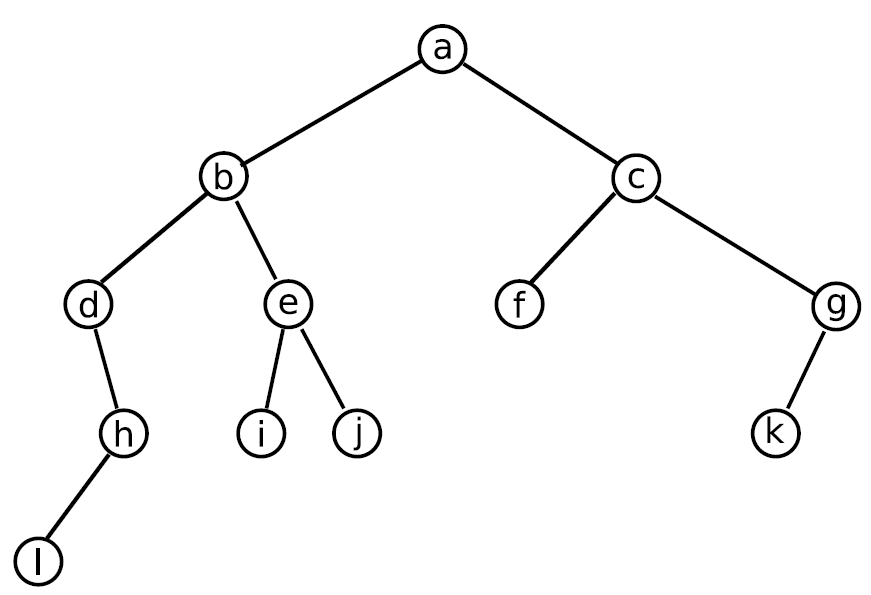
*Povodny graf a jeho indukovany podgraf*

Reprezentacia korenoveho stromu polom rodicov – pole, v ktorom je pre kazdy vrchol ulozeny len nazov  
jeho rodica.

**

Priechod binarnym stromom:  
1. Priechod po urovniach  
2. Priechod po podstromoch

Priechod po urovniach:  
Vloz koren do fronty.  
Dokial je fronta neprazdna:  
 Odstran jej prvy vrchol a preved akciu.  
 Vloz do fronty jeho potomkov v danom poradi.

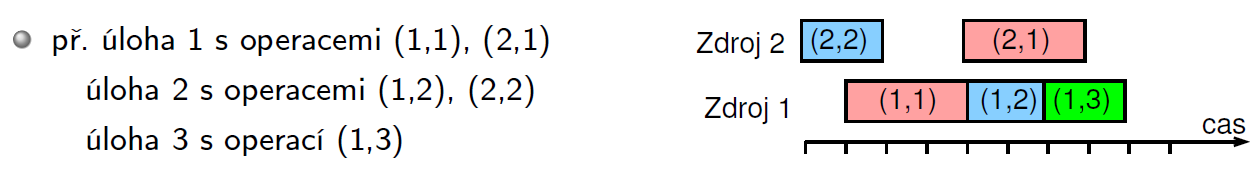
Priechod po podstromoch:  
Preved akciu v koreni stromu.  
Spusti algoritmus priechodu v lavom podstrome.  
Spusti algoritmus priechodu v pravom podstrome.

Priechod stromom po urovniach: a,b,c,d,e,f,g,h,i,j,k,l  
Priechod stromom po podstromoch: a,b,d,h,l,e,i,j,c,f,g,k

Rozvrhovanie a planovanie(scheduling):  
Zdroj/stroj: -kapacita, -dostupnost v case, -rychlost  
Uloha/aktivita: -najskorsi startovny cas, -najneskorsi koncovy cas, -doba trvania, -pocet zdrojov,  
 -alternativne zdroje  
Rozvrhovanie: -optimalna alokacia/priradenie zdrojov v case mnozine uloh: --obmedzene mnozstvo  
 zdrojov, --maximalizacia zisku za danych obmedzeni

Stroje i=1,…,m  
Ulohy j=1,…,n

Sekvencna uloha: uloha je spracovavana na jednom stroji  
Paralelna uloha: uloha je spracovavana na niekolkych strojoch, vsetky operacie su spracovavane zaroven

Operacia(i,j) – prevadzanie ulohy j na stroji i  
 -uloha sa moze skladat z viacerych operacii  
 -typicky sa operacie ulohy spracovavaju na strojoch postupne bez prekrytia  


Parametre ulohy  
1. Staticke parametre  
 -doba trvania pj (pij): doba trvania ulohy j (na stroji i)  
 -termin dostupnosti rj: najblizsie, kedy moze byt uloha j vykonana  
 -termin dokoncenia dj: cas, dokedy najneskor musi byt uloha j dokoncena  
 -vaha wj: dolezitost ulohy j relativne k ostatnym uloham v systeme

2. Dynamicke parametre  
 -cas startu ulohy Sj (Sij): cas, kedy zacne vykonavanie ulohy j (na stroji i)  
 -cas konca ulohy Cj (Cij): cas, kedy je dokoncena uloha j (na stroji i)

Grahamova klasifikacia α | β | γ : pouziva sa pre popis rozvrhovacich problemov  
α: charakteristiky stroja – popisuje sposob alokacie uloh na stroje  
β: charakteristiky uloh – popisuje obmedzenia aplikovane na ulohy  
γ: optimalizacne kriteria

**Stroje(α)**

Zakladne stroje:  
1. Jeden stroj  
2. Identicke paralelne stroje Pm: m identickych paralelne zapojenych strojov, uloha je dana jedinou   
operaciou, uloha moze byt prevadzana na lubovolnom z m strojov

Stroje s roznou rychlostou:  
1. Paralelne stroje s roznou rychlostou Qm: doba trvania ulohy j na stroji i priamo zavisla na jeho  
rychlosti vi, pij=pj/vi, prikladom moze byt niekolko pocitacov s roznou rychlostou procesorov

2. Nezavisle paralelne stroje s roznou rychlostou Rm: stroje maju roznu rychlost pre rozne ulohy,   
stroj i spracovava ulohu j rychlostou vij, pij=pj/vij, prikladom moze byt vektorovy pocitac, ktory pocita  
vektorove ulohy rychlejsie ako klasicke PC

Multi-operacne problemy: jedna uloha je vykonavana postupne na niekolkych strojoch, uloha  
j sa sklada z niekolkych operacii (i,j), operacia (i,j) je vykonana na stroji i za dobu pij  
Pr: uloha j vykonavana 4 operaciami: (1,j),(2,j),(3,j),(4,j)

Open shop Om: multi-operacny problem s m strojmi

Job shop Jm: multi-operacny problem s m strojmi, poradie vykonavania operacii je predom urcene  
Pr: (i,j) -> (k,j) urcuje ze uloha j sa ma vykonat skor na stroji i ako na stroji k

Precedencne podmienky: uloha moze byt vykonavana az po skonceni dalsich uloh  
Pr: pre ulohy a,b piseme: a -> b: Sa + pa <= Sb

Vhodnost stroja: podmnozina strojov Mj, na ktorych sa da dana uloha vykonavat  
Pr: uloha moze byt vykonavana len na tych strojoch v sieti, kde su dostupne data

**Obmedzenia(β)**

Doprava a komunikacia:

-doba nutna na prepravu ulohy j medzi dvoma strojmi k a l  
 -tjkl: doba na prepravu ulohy j zo stroja k na stroj l  
 -tkl: doba na prepravu medzi strojmi k a l nezavisle na ulohe  
 -tj: doba na prepravu ulohy, nezavisle na strojoch

-obmedzenie na prenasane mnozstvo a moznu dobu prepravy

-obmedzenie na priepustnost(kapacitu) linky

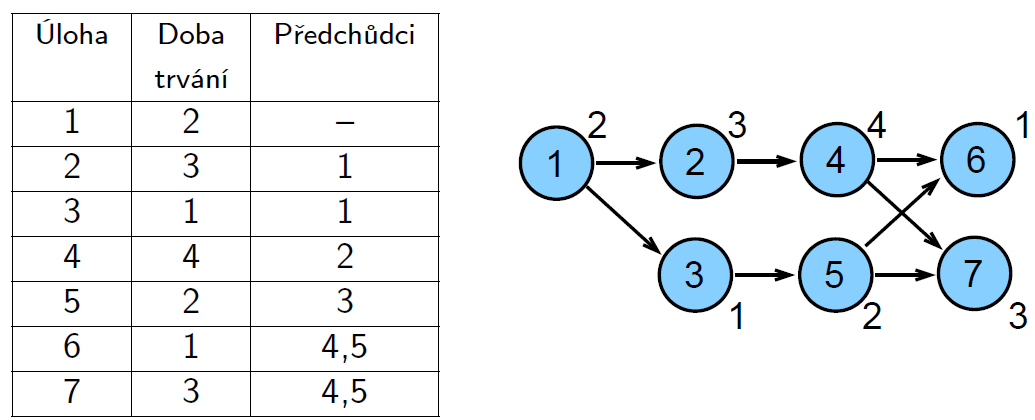
-obmedzenie na vzdialenost uzlov pre prenos

**Optimalizacia(γ)**

Makespan: maximalny cas konca uloh: Cmax = max (C1,...,Cn)  
Pr: Cmax=max{1,2,3,4,5,8,7,9}=9  
minimalizacia makespan maximalizuje vykon, zaistuje rovnomerne zatazenie strojov

Oneskorenie(Lateness) ulohy j: Lj=Cj – dj  
maximalne oneskorenie: Lmax = max (L1,…,Ln)

Nezaporne oneskorenie (tardiness) ulohy j: Tj = max (Cj – dj, 0)  
Ciel: minimalizacia celkoveho oneskorenia, minimalizacia celkoveho vazeneho oneskorenia  
Celkove oneskorenie: T1+T2+…+Tn  
Celkove vazene oneskorenie: w1T1+w2T2+…+wnTn

Graficka reprezentacia precedencnych obmedzeni  
-pomocou orientovaneho acyklickeho vrcholovo ohodnoteneho grafu  


Disjunktivna grafova reprezentacia

Graf G = (N, A∪ B ∪ P)  
- uzly zodpovedaju operaciam N = {(i,j)} ∪ {U,V}  
- dva pomocne uzly U a V reprezentuju zdroj a stok  
- konjunktivne hrany A reprezentuju poradie operacii ulohy  
 Pr: (i,j) -> (k,j) ∊ A, potom operacia (i,j) predchadza operacii (k,j)  
- disjunktivne hrany B reprezentuju konflikty na strojoch  
 Pr: dve operacie (i,j) a (i,l) su spojene dvoma opacne orientovanymi hranami  
- pomocne hrany P: hrany z U ku vsetkym prvym operaciam ulohy, hrany zo vsetkych poslednych  
 operacii do V

## Vyber: Podmnozina D ⊂ B, ak D obsahuje z kazdeho paru disjunktivnych hran prave jednu Vyber D je splnitelny, ak vysledny orientovany graf G(D) = (N, A∪ D ∪ P) je acyklicky -jedna sa o graf s pomocnymi, konjunktivnymi a vybranymi disjunktivnymi hranami

## Planovanie projektu

## -precedencne podmienky, paralelny stroj s neobmedzenym poctom strojov, minimalizacia makespan metodou kritickej cesty, kriticka cesta urcuje makespan

## Variabilna doba trvania: -doba trvania ulohy spojena s cenou vykonavania -optimalizacia: kompromis medzi cenou za ukoncenie projektu a cenou za skratenie dlzky uloh

## Metoda kritickej cesty (critical path method CPM)

## dopredna procedura -start v case 0 -vypocet najskorsieho startoveho casu kazdej ulohy -cas dokoncenia poslednej ulohy je makespan

## spatna procedura: -start v case rovnom makespan -vypocet najneskorsieho startoveho casu, aby bol realizovany tento makespan

## Uloha s rezervou (slack job) -jej startovny cas moze byt odlozeny bez toho, aby bol navyseny makespan -uloha j, ktorej najskorsi startovny cas Sj’ je mensi ako najneskorsi startovny cas Sj’’ -rezerva ulohy: Sj’’ – Sj’

## Kriticka uloha -uloha, ktora nemoze byt odlozena -jej najskorsi startovny cas je rovny najneskorsiemu startovnemu casu

## Kriticka cesta -retazec uloh zacinajucich v case 0 a konciacich v case Cmax -v grafe moze existovat viac kritickych uloh a mozu sa ciastocne prekryvat Graf kritickych ciest: podgraf dany mnozinou kritickych uloh a kritickych ciest

## Marginalna cena: cena za skratenie doby trvania ulohy o 1 casovu jednotku

## Fixne rezijne naklady c0: na casovu doby vykonavania projektu Cena F(pj) za vykonavanie projektu: pri dobe prevadzania uloh pj, urcena ako sucet: ceny za prevadzanie vsetkych uloh + fixne rezijne naklady

## Orientovany graf G = (V,E) Pociatocny uzol: zdroj s ∊ V Koncovy uzol: stok t ∊ V

## Rez: mnozina uzlov V’, ktorych zmazanim z grafu sa rozpoji zdroj a stok, tj v grafe G’=(V-V’, E-E’) neexistuje orientovana cesta z s do t Minimalny rez: rez U taky, ze neexistuje rez W ⊂ U, tj vratenie lubovolneho uzlu z U do grafu znovu spoji zdroj a stok

## Chromaticke cislo grafu: minimalny pocet farieb n nutny k zafarbeniu grafu tak, aby ziadne dva susedne vrcholy neboli zafarbene rovnakou farbou

## Stupen vrcholu: pocet hran spojenych s vrcholom Uroven saturacie: pocet roznych farieb spojenych s vrcholom

## Priradenie miestnosti vrchol: predmet hrana: medzi predmetmi, vyzadujucimi rovnaky cas (prienik casovych bodov) susediace vrcholy musia mat rozne farby, pretoze vyzaduju rovnaky cas

## Rezervacny problem vrchol: rezervacia hrana: medzi rezervaciami, ktorych casy sa prekryvaju (prienik intervalov) susediace vrcholy musia mat rozne farby, pretoze sa prekryvaju v case kolko zdrojov (izba v hoteli…) je potrebnych pre splnenie rezervacii = chromaticke cislo grafu

## Rozvrhovanie operatorov vrchol: uloha hrana: medzi ulohami vyzadujucimi rovnakeho operatora susediace vrcholy musia mat rozne farby, pretoze vyzaduju rovnakeho operatora makespan = chromaticke cislo grafu

## Dlzka cesty v neohodnotenom grafe: pocet hran na tejto ceste Dlzka cesty v ohodnotenom grafe: sucet ohodnoteni hran na tejto ceste Vzdialenost δ(u,v) vrcholov u a v v grafe, je dlzka najkratsej cesty z u do v, pre orientovany graf nemusi byt vzdialenost symetricka

## Smerovanie(Routing): proces najdenia cesty medzi dvoma komunikujucimi uzlami, spravidla sa smerovac nezaobera celou cestou paketu, ale len o jeden krok – komu dalsiemu paket poslat Cesta musi splnat nejake obmedzujuce podmienky Ovplyvnujuce faktory: -staticke: topologia siete -dynamicke: zataz siete

## Smerovanie: - spolocna cinnost smerovacov(globalna) - process najdenia/vytvorenia a udrzby smerovacich tabuliek

## Zasielanie(Forwarding): - kazdy smerovac samostatne - predstavuje process priechodu paketov smerovacom -zaslanie paketu na vybrane rozhranie smerovaca(podla adresy) - vyzaduje pristup k smerovacej tabulke

## Cena komunikacie/linky: metrika 1.Vsetky linky maju rovnaku cenu => cena/metrika cesty = sucet skokov zo zdroja do ciela 2. Cena linky = prevratena hodnota kapacity (1/prenosova\_kapacita) 3. Cena linky = oneskorenie linky 4. Cena linky = vyuzitie linky 5. Cena linky = realna cena za vyuzitie linky

## Distribuovane smerovanie

## Distance Vector (DV) – Bellman-Fordov algoritmus susedne smerovace si v pravidelnych intervaloch alebo pri zmene v sieti vymienaju cele svoje smerovacie tabulky, na zaklade prijatych updatov si doplnaju nove informacie a inkrementuju svoje distance vector cislo (metriku udavajucu pocet hopov k danej sieti)

## VSETKY INFORMACIE LEN SVOJIM SUSEDOM

## Zaznamy v smerovacej tabulke su usporiadane trojice (N,G,D) N – cielova siet G – adresa nasledujuceho smerovaca D – vzdialenost/metrika do cielovej siete

## Kvoli pomalej konvergencii vznikaju v smerovacich tabulkach nespravne udaje a slucky Riesenie: delenie horizontu: smerovac neposkytuje informacie o ceste uzlu, od ktoreho sa o tejto ceste dozvedel

## Link State (LS) jednotlive smerovace si zasielaju len informacie o stavoch priamo pripojenych liniek, udrzuju si kompletne informacie o vsetkych linkach v sieti INFORMACIE O SVOJICH SUSEDOCH VSETKYM

## Djikstrov algoritmus -algoritmus hladania najkratsej cesty -najde najkratsiu cestu z jedneho vrcholu do vsetkych ostatnych => musi prejst vsetky vrcholy -funguje pre orientovany aj neorientovany graf -okrem nezapornych cyklov vyzaduje nezaporne ohodnotenie vsetkych hran -linearna pamatova zlozitost -casova zlozitost zalezi na pouzitej datovej structure

## Bellman-Ford algoritmus -rovnako ako Djikstrov vypocita vzdialenost vsetkych vrcholov z jedneho zdroja -graf moze obsahovat aj zaporne ohodnotene hrany -detekuje cykly s celkovym zapornym ohodnotenim -vyssia casova zlozitost ako djikstrov algoritmus

## OSPF (Open Shortest Path First) -najpouzivanejsi LS protocol metrika: cislo(1 – 65535), priradene ku kazdemu rozhraniu smerovaca, cim mensie cislo => tym lepsia metrika, standardne je metrika 100000000/bandwidth v bps

## Kvalita sluzby Zakladne parametre sietovych tokov: 1. Spolahlivost(reliability): poziadavka plnej spolahlivosti vs. tolerancie definovanej stratovosti 2. Oneskorenie(latency,delay) 3. Rozptyl oneskorenia(jitter) 4. Prenosova kapacita(bandwidth)

## A\* algoritmus -upraveny Djikstrov algoritmus -pouziva sa hlavne pre najdenie cesty do jedneho vrcholu -okrem dlzky najkratsej cesty do kazdeho vrcholu, bere pri vyhladavani v uvahu aj heuristicky odhad jeho vzdialenosti od ciela -kazdej ceste je priradeny heuristicky odhad dlzky ako sucet ohodnoceni hran cesty a heuristicke ohodnotenie koncoveho vrcholu -do prioritnej fronty niesu ukladane vrcholy grafu, ale cesty, najvyssiu prioritu ma cesta s najnizsim heuristickym ohodnotenim -casova zlozitost zalezi od heuristiky, najhorsie: exponencialna, najlepsie: polynomialna voci dlzke optimalnej cesty, pametova zlozitost najhorsie moze byt exponencialna

## Floyd-Warshallov algoritmus -vypocitava najkratsiu vzdialenost medzi vsetkymi dvojicami vrcholov v grafe -graf moze obsahovat zaporne ohodnotene hrany, cykly s celk. Zapornym ohodnotenim vedu k chybe -medzi kazdymi dvoma dvojicami vrcholov postupne vylepsuje najkratsiu znamu vzdialenost -casova zlozitost algoritmu: O(n^3), pametova zlozitost: O(n^2)

## Planovanie na pocitacovej sieti

## Planovanie komunikacie sucasne beziacich uloh: -planovanie strojov pre ulohy -planovanie sucasne prebiehajucej komunikacie

## Planovanie sucasne prebiehajucich prenosov: -planovanie vsetkych komunikacii pre aktualny okamih -planovanie strojov neuvazovane

## Planovanie uloh s precedentmi: -planovanie strojov pre ulohy -planovanie prenosu dat po ukonceni ulohy inej ulohe (uloham)

## Rozdelenie grafu (Graph partitioning) G = (V,E) na V = V1 ∪… ∪ Vm tak, ze je -V1 ∩ … ∩ Vm = {} -G1 = (V1, E1),…, Gm = (Vm,Em) -Ei tvorene hranami, ktorej oba vrcholy patria do Vi

## Bisekcia grafu: z grafu G = (V,E) sa vytvoria dva podgrafy (V1,E1) a (V2,E2) tak, ze: -V=V1∪V2, V1∩V2={} -problem optimalneho rozdelenia grafu je NP-uplny

## Rozdelenie grafu so suradnicami vrcholov: -kazdy vrchol ma suradnice v priestore -graf sa rozdeli pomocou deliacej priamky, ktora deli vrcholy v priestore na polovice

## Planovanie datovych prenosov 1. Planovanie prenosu v case: je treba uvazovat planovanie zdrojov v case 2. Planovanie sucasne prebiehajucich prenosov: cas nieje uvazovany, vsetko sa planuje na teraz

## Grafova reprezentacia v sieti

## Orientovany graf (V,L) vrcholy: mnozina uzlov v ∊ V hrany: mnozina orientovanych liniek l ∊ L

## Uzol v ∊ V moze mat nasledujuce role: 1. Producent p ∊ P: produkuje data, posiela data prave jednou linkou 2. Konzument c ∊ C: prijima data, prijima data prave jednou linkou 3. Distributor d ∊ D: rozosiela prijate data do niekolkych liniek Distributor moze: 1. Data transformovat, napr: z HDTV videa na HDV MPEG 2 video 2. Data su rovnakeho typu Distributor prijima data najviac jednou linkou, na distributorovi nevznikaju ziadne data inlinks(d): linky vstupujuce do uzlu distributora outlinks(d): linky vystupujuce z uzlu distributora

## Linka l ∊ L ma urcene: 1. Pociatocny uzol begin(l) 2. Koncovy uzol end(l) 3. Oneskorenie latency(l) 4. Kapacitu capacity(l)

## Stream s ∊ S: datovy tok od producenta ku konzumentom -bandwidth(s): udava pozadovanu sirku pasma datoveho toku stream -prijem dat streamu je pozadovany mnozinou consumers(s) ⊆ C -data streamu vysiela len jeden producent => pre kazdy stream je presne urceny producent

## Distribucny strom pre kazdy stream s -distribucny strom ako podgraf grafu (V,L) -koren stromu: producent(s) streamu -listy stromu: uzly s konzumentami(s) streamu -vnutorne uzly: uzly s distributormi

## Modely pre planovanie datovych prenosov 1. Model zalozeny na linkach (link-based model) 2. Model zalozeny na cestach (path-based model) 3. Model zalozeny na uzloch (node-based model)

## 1.Model zalozeny na linkach: pre kazdu poziadavku streamu s: premenna Xs,l pre kazdy s a linku l Xs,l = 1, ak s prechadza po l, Xs,l = 0, inak

## 2.Model zalozeny na cestach: pre kazdu poziadavku streamu s premenna Xs,p pre kazdu moznu cestu p medzi producentom a konzumentom, Xs,p=1, ak poziadavka s ide po ceste p, Xs,p=0, inak

## 3.Model zalozeny na uzloch: pre kazdu poziadavku streamu s premenna Xs,v pre kazdy vrchol Xs,v=1, ak je uzol pouzity pri poziadavke s, Xs,v=0, inak

## Vsetky cykly medzi distributormi je nutne zakazat, v strome s k uzlami, treba mat najviac k-1 hran, pridanim dalsej hrany vznikne cyklus

## Planovanie uloh a datovych prenosov v priebehu casu

## Priklady aplikacii: 1. Planovanie vypoctovych uloh na pocitace -planovanie uloh vyzadujucich procesory, disky… -precedencne zavislosti medzi ulohami je menej caste

## 2.Planovanie komunikujucich uloh s precedentmi -acyklicky graf precedencnych zavislosti medzi ulohami -prenos dat po skonceni ulohy nasledujucim uloham

## 3.Planovanie acyklickych workflows

## Algoritmy: 1. Optimalne(polynomialna zlozitost) -pouzitelne len pre specializovane problemy, obtiazna rozsiritelnost na obecne problemy

## 2.Heuristicke -planovanie zoznamom(list scheduling) -heuristiky mapovania(mapping heuristics) -zhlukovacie heuristiky(clustering heuristics)

## Ulohy a komunikacia: Referencna doba trvania pj ulohy j Precedencne obmedzenie j1 -> j2 Komunikacia: sizej1,j2: velkost prenesenych dat medzi ulohami j1 a j2 pokial existuje j1 -> j2: sizej1,j2 >=0, inak sizej1,j2=0

## Hranovo a vrcholovo ohodnoteny acyklicky orientovany graf -vrcholy = ulohy -orientovane hrany = precedencne vztahy medzi ulohami -ohodnotenie hrany = velkost dat na komunikaciu -ohodnotenie vrcholu = referencna doba trvania

## transratei1,i2 = prenosova rychlost medzi susednymi strojmi i1,i2 setupmesgi = doba na poslanie spravy strojom i komunikacne oneskorenie c(j1,j2,i1,i2) pre poslanie dat z ulohy j1 ulohe j2 zo stroja i1 na stroj i2 speedi = rychlost spracovania strojom i setuptaski = doba na nastartovanie ulohy na stroji i dobra prevadzania pij ulohy j na stroji i = pij=pj/speedi+setuptaski

## Planovanie zoznamom(list scheduling) -jednoduchy efektivny algoritmus -zalozene na zoradeni uloh v prioritnej fronte -zlozitost algoritmu zavisi na vypocte: 1. Priority urcujucej poradie spustenia uloh, 2. Metody vyberu stroja => algoritmus je parametrizovany: 1. Vypoctom priority P, 2. Vyberom stroja M

## Heuristiky mapovania (mapping heuristics) -rozsirenie metod planovania zoznamom -pre navrh heuristik su uvazovane dalsie realne parametre: topologia siete, rychlost procesoru, prenosova rychlost -konstruovana a udrziavana smerovacia tabulka obsahujuca: -odhadovane hodnoty pij a c(j1,j2,i1,i2) -pre kazdy processor: pocet hopov, preferovana linka pre dany ciel, oneskorenie pretazenim -koncovy cas ulohy na stroji je vypocitany z: rychlost procesora, prenosova rychlost, vybrana linka, pocet hopov, oneskorenie dane zatazenim linky -ulohy planovane podla najvyssej urovne, v pripade nejednoznacnosti je vybrana uloha s vacsim mnozstvom priamych nasledovnikov -akonahle su dokonceny vsetci predchodcovia ulohy, tak je uloha naplanovana na stroj, kde skonci najskor

## Zhlukovacie heuristiky (clustering heuristics) -primarne riesi priradenie uloh na stroje -nasledne riesi naplanovanie uloh v case na vybrany stroj -Zhluk: mnozina uloh, ktora bude vykonavana na rovnakom stroji

## Nech rez C deli vrcholy grafu na mnoziny P, P’. Hrany (u,v), ktorych jeden vrchol patri do P a druhy nie, krizuju rez C. WG(P) = mnozina vsetkych hran, ktorych jeden vrchol patri do P a druhy nie.

## Bipartitny graf: graf G, ktoreho mnozina vrcholov je disjunktivnym zjednotenim mnozin S a T a plati, E(G)=WG(S). Mnoziny S a T nazyvame stranami bipartitneho grafu. Uplny bipartitny graf: bipartitny graf, ktoreho kazda dvojica vrcholov (s,t), je spojena jednou hranou.

## Vaha rezu: v hranovo neohodnotenom grafe je to pocet hran, ktore krizuju rez, v hranovo ohodnotenom grafe je to sucet ohodnoteni hran, ktore krizuju rez. Minimalny rez: rez v grafe s minimalnou vahou, (najdenie v polynomialnom case) Maximalny rez: rez v grafe s maximalnou vahou (najdenie je NP-uplny problem)

## Siet = orientovany, hranovo ohodnoteny graf G = (V,E)

## Tok v sieti = ohodnotenie hran realnymi cislami f: E(G) -> R, ktore pre kazdy vrchol v splna Kirchhoffov zakon: kolko do vrcholu vstupuje, tolko z neho aj vystupuje

## Cirkulacia = pokial kirchhoffov zakon plati pre vsetky vrcholy

## Tok od zdroja k spotrebicu = dva vrcholy nesplnaju kirchhoffov zakon, v zdroji tok vznika, a v spotrebici tok zanika, pridanim navratovej hrany medzi zdrojom a spotrebicom vznikne cirkulacia

## Pripustny tok = tok, ktory splna l(e) <= f(e) <= c(e) pre vsetky hrany e, pricom: l(e) = dolne obmedzenie toku, c(e) = horne obmedzenie toku

Rezidualna kapacita hrany = cislo c(e) – f(e) tj. Rozdiel kapacity hrany a aktualneho toku  
Rezidualna siet = rezidualne kapacity hran

Velkost toku od zdroja k spotrebicu = mnozstvo toku, ktore vznika na zdroji s => sucet tokov vtekajucich do zdroja minus sucet tokov vytekajucich zo zdroja

Velkost toku cez rez = zdroj nalezi do mnoziny P a spotrebic do mnoziny P’ => Rez oddeluje zdroj  
a spotrebic => velkost toku cez rez je rozdiel medzi velkostou toku na hranach veducich z mnoziny P  
a velkostou toku na hranach veducich do mnoziny P.

Kapacita rezu oddelujuceho zdroj a spotrebic = maximalny tok, ktory moze tymto rezom pretekat, je   
urceny ako sucet kapacit vsetkych hran, pretinajucich rez v smere od zdroja k spotrebicu minus sucet  
minimalnych kapacit hran opacne orientovanych

Algoritmy pre hladanie maximalneho toku:  
1. Brutalna sila  
2. Zlepsujuca cesta  
3. Ford-Fulkersonov algoritmus (pouziva sa pri unicast)  
4. Edmonds-Karpov algoritmus

1.Brutalna sila: generuje postupne vsetky podmnoziny vrcholov, pre kazdu prevedie:  
 -najde vsetky hrany krizujuce rez definovany danou mnozinou vrcholov  
 -scita kapacity hran krizujucich tento rez od zdroja k spotrebicu  
 -casova zlozitost O(2|V|-2|E|)

2.Zlepsujuca cesta:   
Hrana vpred = hrana orientovana v smere priechodu cestou, hrana vzad = proti smere  
Zlepsujuca cesta = neorientovana cesta zo zdroja k spotrebicu, ktorej kazda hrana splna   
f(e) < c(e) pre hranu e vpred a f(e) > l(e) pre hranu e vzad  
Aktualny tok sa da zvysit na hranach vpred a znizit na hranach vzad o hodnotu d > 0  
Kapacita zlepsujucej cesty = maximalna hodnota d, o ktoru sa da tok zmenit

3.Ford-Fulkersonov algoritmus:  
-vyuziva ekvivalenciu medzi maximalitou toku a neexistenciou cesty zo zdroja k spotrebicu v   
rezidualnej sieti  
-hlada zlepsujuce cesty zo zdroja do spotrebica, pokial nejaka existuje  
-pre hladanie pouziva aj zpatne hrany  
-z hran na ceste sa vyberie ta, ktorej rezidualna kapacita je minimalna

4.Edmonds-Karpov algoritmus  
-specializacia Ford-Fulkersonovho algoritmu  
-pre hladanie zlepsujucich ciest ja pouzity priechod do sirky  
-pre potreby priechodu do sirky su dlzky hran povazovane za jednotkove  
-priechod do sirky zabezpeci, ze kazda najdena zlepsujuca cesta je najmenej tak dlha, ako predchodzia  
-maximalna dlzka zlepsujucej cesty je |V|  
-zlozitost algoritmu je O(VE2)

Obmedzenie toku vrcholom = obmedzena velkost toku prechadzajuceho vrcholom

Kazda hrana ma okrem kapacity definovanu aj cenu a(e) jednotkoveho toku  
=> cena toku hranou e = a(e)f(e), celkova cena toku sietou je suma a(e)f(e) pre vsetky hrany

Pripustna cirkulacia: existuje v sieti s obmedzeniami toku l a c, vtedy, ked kazdy rez ma nezapornu kapac.

**Network Coding**

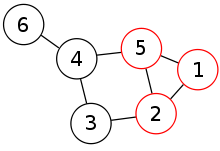
MaxFlow-MinCut: veta o priepustnosti siete, hovori, ze maximalna priepustnost je rovna minimalnemu  
rezu takemu, ze zdroj dat je v T a prijimajuci v T’  
Definicia plati pre: -unicast, -broadcast, -multicast

Kodovanie: kodovanie prichadzajucej informacie (operacia XOR), pouziva sa pri multicast, pretoze   
pre multicast neexistuje efektivny algoritmus pre hladanie maximalnej priepustnosti

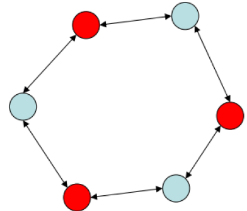
Kodovacia schema: urcuje pre kazdy uzol, ako ma byt vstupna informacia kodovana, musi byt   
vytvorena pred prenosom, ked dojde k zmene topologie, musi byt vypocitana znovu, pracuje  
so zjednodusenym modelom siete (rovnake kapacity liniek, oneskorenie na linkach rovnake…)

Polynomialne algoritmy pre vytvaranie kodovacich schem: LIF, LIFE, LIFE-CYCLE, LIFE\*   
-prechadzaju graf od zdroja k prijemcovi  
-konstruuju kodovacie funkcie pre kazdy uzol tak, aby vysledny system obsahoval dostatocny pocet  
 linearne nezavislych funkcii (inak by neexistovalo riesenie a prijemca by nemohol dekodovat)

Nahodnostne kodovanie: pomocou neho sa da dosiahnut maximalna priepustnost bez znalosti  
topologie  
-uzly v sieti koduju prichadzajuce informacie pomocou nahodne vygenerovanych koeficientov, ktore  
 sa zasielaju spolu s datami  
-dostatocny pocet linearnych kombinacii na dekodovanie na prijimacej stanici: tolko, kolko   
 jednotlivych blokov informacie sa obdrzalo

Clique(Klika) = najvacsi uplny podgraf grafu, ktoreho vsetky vrcholy su spojene hranou so vsetkymi  
ostatnymi vrcholmi

Najvacsia klika grafu je (1,2,5)

N-klika = podgraf grafu, v ktorom cesta medzi dvoma uzlami neni vacsia ako N, n-kliky mozu byt  
nekompaktne = tj. vrcholy mozu patrit do rovnakej n-kliky aj ked spolu priamo nesusedia

N-klan: podobne n-klike, ale cesta nesmie viest cez vrcholy, ktore niesu sucastou N-klanu  
prislusnost do N-klanu je dana cestami dlhymi najviac N hran, bez pouzitia uzlov mimo N-klan

K-plex: mnozina uzlov, v ktorej vsetky uzly susedia s ostatnymi s vynimkou najviac K-uzlov  
Pr: 2-plex => K=2 => vsetky uzly v 2-plexe su spojene s aspon N-2 dalsimi uzlami

Kompletnost grafu = (<pocet uzlov grafu> - <pocet izolovanych uzlov> / <pocet uzlov grafu>)

Lokalna centralita  
-stupen vrcholu grafu  
-nemoze byt kriteriom pre zrovnavanie rozne velkych grafov

Relativna lokalna centralita  
-pomer skutocneho poctu hran k maximalnemu poctu hran  
-da sa pouzit pre zrovnavanie rozne velkych grafov, neda sa pouzit pre grafy roznych kategorii relacii  
Pr: uzol stupna 25 ma v grafe so 100 vrcholmi relativnu lokalnu centralitu = 25/100 = 0,25

Globalna centralita  
-vzdialenost (dlzka cesty) medzi roznymi vrcholmi grafu  
-vrchol je globalne centralny, ak suma dlzok vsetkych jeho ciest k vacsine ostatnych uzlov je najmensia  
 tj. ma najkratsie cesty k vacsine ostatnych uzlov  
-uzitocnost: udava ocakavany cas dorucenia lubovolnej informacie prenasanej po sieti

Stupen centrality  
-udava mieru vystavenia informacii prechadzajucich grafom (centralny akter bude mat presnejsie   
 informacie ako aktery na odlahlych koncoch grafu)

Dosah centrality  
-mnozina vrcholov, ktore su vzajomne dosiahnutelne v najviac k-krokoch  
-Pre k=1=stupen centrality

Centralizacia: rozdiel medzi lokalnou centralitou kazdeho uzlu a globalne centralnym uzlom,   
udava mieru dominancie jedneho alebo viacerych akterov na sieti

Hustota: pomer poctu hran medzi uzlami k maximalnemu poctu hran v uplnom grafe

Kliky,N-kliky,N-klany: uzitocne pre identifikaciu skupin v sieti

**Planovanie v telekomunikacnych sietach**

Optimalizacia  
-kombinatoricke problemy (rozlozenie stanic GSM, rozdelenie frekvencie pri wifi…)  
-NP – tazke ulohy

Heuristiky  
-priblizne riesenie NP-tazkych problemov  
-prehladavanie: 1. Lokalne prehladavanie a jeho varianty, 2. Evolucne prehladavanie  
-standardne techniky: 1. Linearne programovanie, 2. Dynamicke programovanie atd…  
-jednoduchsie pouzitie a vacsia sanca na najdenie aspon jedneho riesenia

Lokalne prehladavanie: zacne s nejakym riesenim a to “vylepsuje” prehladavanim jeho okolia  
-potrebuje definovat okolie  
-zmena stavu/mutacia: pridanie alebo ubranie vrcholu alebo hrany, posun hrany  
-porovnava cenu aktualneho riesenia s novym najdenym riesenim, ak cena noveho je mensia, nahradi  
 aktualne riesenie novym najdenym

Uviaznutie lokalneho prehladavania: v pripade, ze je najdene lokalne minimum => z neho sa neda  
dostat bez porusenia pravidla nezhorsovania cesty

Pouzivane pristupy pri uviaznuti:  
1. Simulovane zihanie: s urcitou pravdepodobnostou akceptujeme aj horsie riesenie  
2. Tabu prehladavanie: zohladnuje typ mutacie, nie len cenu

Simulovane zihanie  
-zavadza teplotu T (pri vyssej teplote je material “tvarny”)  
-vyssia teplota = vyssia pravdepodobnost akceptovania horsieho riesenia  
-spocita funkciu F(f(c),f(c’),T)  
-vygeneruje nahodne cislo r ∊ (0,1)  
-pokial F > r => konfiguracia c’ je akceptovana  
-teplota klesa s poctom iteracii

Tabu prehladavanie  
-pamata si predchadzajuce zmeny  
-zavadza tabu(zakazane) zmeny  
-vybere najlepsiu z povolenych zmien (nemusi byt nutne najlepsia konfiguracia)  
-podstatny vyber kriterii, podla ktorych sa zaraduje do zakazanych (tabu) zoznamov

Evolucne prehladavanie  
-pri hladani novej mutacie, sa pouziva cela populacia (prehladava sa paralelne viacero okoli)  
-rekombinacia: dva alebo viac “rodicov”, vhodnou operaciou sa spoja ich vlastnosti, vysledok  
 sa nazyva recombinant  
-crossover: mame stav popisany linearnym vektorom, vysledna konfiguracia nahodny vyber hodnot   
 z “rodicovskych” konfiguracii (vektorov)  
-rekombinant sa moze aj o 50% lisit od ktorehokolvek “rodica”

Hierarchia prvkov mobilnej siete  
Mobilne stanice (MS) komunikuju bezdrotovo so zakladnovymi stanicami (BTS), tie su kontrolovane  
riadiacimi stanicami (BSC), ktore su zdruzene pod prepinacimi stanicami (MSC), samotne prepinacie  
stanice tvoria chrbticu mobilnej siete, pretoze zodpovedaju autentizacii, uctovnictvu, udrzbe databaz…

Technicke parametre:  
-Nk=int(ΔF/Fk) – pocet frekvencnych kanalov, ktore mozu byt pouzite, Fk=sirka jedneho pasma  
-ns=int(Nk/C x M) – pocet frekvencii potrebnych pre obsluhu uzivatelov (M) v jednom sektore   
 kazdej bunky (C je pocet sektorov)   
-NaBTS = M x int(A / β) – pocet uzivatelov jednej bazovej stanice  
-NBTS = int(Na/NaBTS) – nevyhnutny minimalny pocet zakladnovych stanic pre pokrytie danej  
 geografickej oblasti, (Na = pocet uzivatelov, ktorych ma jedna zakladnova stanica obsluhovat)  
-R=sqrt((1,21 x S0) / (πNBTS)) – polomer bunky(za predpokladu rovnomerneho pokrytia)  
-D = R sqrt(3C) – bezpecna vzdialenost medzi zakladnovymi stanicami pouzivajucimi rovnaku frekvenciu

Alokacia kanalov(frekvencii)  
-k dispozicii je obmedzeny pocet kanalov/frekvencii  
-zakladna idea mobilnej siete: znovupouzitie frekvencie v geograficky oddelenych castiach siete  
-Bunka = cast siete s mnozinou pridelenych frekvencii, cele spektrum je pridelene clusteru buniek  
 obmedzeny pocet buniek v clustery: N = i2 + ij + j2, kde i a j su cele cisla (N je teda 1,3,4,7…)  
Podmienky alokacie:  
1. Interferencia frekvencii medzi bunkami  
2. Interferencia frekvencii v bunke  
3. Poziadavky bunky na pocet kanalov  
Problem pridelenia kanalu ⬄ zobecnenemu problem farbenia grafu

Dominujuca mnozina grafu je taka mnozina vrcholov, ze vsetky ostatne vrcholy su spojene s   
aspon jednym vrcholom dominujucej mnoziny.

**Topologia internet**

Uzly v internete sa spajaju:  
1. Vo vnutri autonomnych jednotiek: podla potrieb spravcu  
2. Medzi autonomnymi jednotkami: podla vzajomnej dohody bez centralnej authority  
 -za poplatok (pripojenie k ISP medzi mensim a vacsim ISP)  
 -vzajomne vyhodne(vymena dat, zdielanie kapacit,atd…)

Ziadna centralna autorita neschvaluje ani neeviduje budovanie sieti

Modely topologie internetu: Popis Internetu ako grafu  
-uzly: jednotlive zariadenia, routery, autonomne systemy  
-prepojenie na L1-L3 ISO OSI: ohodnotenie kapacitou, oneskorenim, jitterom, velkostou bufferov…

Metody vyskumu topologie internetu  
1. Meranie skutocnej topologie  
 -analyza traceroute, BGP  
 -projekty Rocketfuel, Skitter, Archipelago  
 -meracie uzly, aktivna vzajomna komunikacia

2. Teoreticka praca  
 -hladanie charakteristickych vlastnosti a javov v nameranych datach  
 -navrh algoritmov pre generovanie nahodnych sieti s najdenymi charakteristikami

Erdos-Renyi model  
-obecny model nahodnych grafov, bez prisposobenia sietam  
-model G(n,p), n=pevny pocet vrcholov, p=pravdepodobnost vzniku kazdej hrany, vrcholy su nahodne  
 spajane, kazda dvojica nezavisle na sebe  
-priemerne (n nad 2) \* p hran  
-vsetky hrany rovnako pravdepodobone – nerealne => castejsie sa spajaju blizke uzly alebo uzly  
 podobnej kapacity a vyznamu

Waxmanov model  
-prvy model nahodneho generovania grafu s prihliadnutim na siet  
-oproti obecnemu modelu nieje vznik vsetkych hran rovnako pravdepodobny

Transit-Stub model  
-Kritika waxmanovho modelu:   
 grafy sa vizualne nepodobaju realnym sietam  
 -uzly nemaju ziadnu hierarchiu  
 -chyba obvykla chrbtova siet  
 -objavuju sa nelogicke linky na dlhe vzdialenosti

Grafy niesu spojite

-Zavadza tri stupne hierarchie: 1.Tranzitna(Transit) AS, 2. Koncova (Stub) AS, 3. Lokalne siete

Power law: statisticke oznacenie pre vztah dvoch velicin, kde zavisla premenna rastie alebo klesa  
s mocninou nezavislej premennej

Power law v Internete – bezskalove siete

1.Stupen vrcholu vs frekvencia  
dv – stupen vrcholu  
fv – frekvencia vrcholov stupna dv  
-vrcholy vysokeho stupna su vynimocne, frekvencia vrcholu rastie s klesajucim stupnom   
=> fv ~ (-dv)2,2

2.Pocet hopov vs pocet dvojic vrcholov v nanajvys tejto vzdialenosti  
P(h) – pocet dvojic vrcholov vo vzdialenosti nanajvys h (merane poctom hopov)  
Pocet dvojic vrcholov, ktore su vzajomne dosiahnutelne v h hopoch rastie s h   
=> P(h) ~ h4,7

Model Barabasi-Albert  
-zohladnuje 2 aspekty realnych sieti: 1.Siet od svojho vzniku stale rastie, 2. Preferenciu pripojenia  
 k vrcholom vyssieho stupna  
-prirodzene vytvara suvisle bezskalove grafy, fv ~ (-dv)2,9

Kritika bezskalovych sieti: vznik hubov  
-kriticke uzly siete, prechadza nimi vacsina provozu, tvoria uzke hrdlo  
-spajaju siet dohromady, ich vypadok vedie k zasadnemu obmedzeniu konektivity

L(g) metrika  
normalizovana metrika: L(g) = (l(g) – lmin)/(lmax – lmin), L(g) ∊ [0,1]  
l(g) = ∑ didj , kde (i,j) ∊ E(g)  
-uvazujeme jednu postupnost stupnov (d1,d2,…,dn)  
-mnozina grafov G, vsetky s touto postupnostou stupnov  
 lmax = max {l(g) : g ∊ G}  
 lmin = min {l(g) : g ∊ G}

Pristup First-Principle

Technologicke obmedzenia  
-routre dostupne na trhu  
-kompromis medzi celkovou priepustnostou a stupnom vrcholu  
-so stupnom neklesa len priepustnost pre kazde pripojenie, ale priepustnost celkova

Ekonomicke obmedzenia  
-provoz fyzickych liniek je drahy  
-snaha o maximalnu agregaciu do liniek vyssich kapacit co najblizsie koncovym uzlom  
-chrbtica tvori relativne malo dlhych liniek vysokych kapacit

HOT – Heuristicky Optimalne Topologie  
-obvykla predstava prevadzkovatelov o dobrej topologii siete

Vznik HOT grafu  
-vyber 50 centralnych uzlov nizsieho stupna d\o jadra  
-ich susedia vyssich stupnov ako brany  
-redistribucia hran medzi branami a jadrom

dK-rozdelenie: pravdepodobnostne rozdelenie na podgrafoch velkosti:  
 0K: priemerny stupen vrcholu  
 1K: rozdelenie stupnov vrcholu  
 2K: pravdepodobnost spojenia vrcholov o danych stupnoch  
 3K: rozdelenie podgrafov o 3 vrcholoch

dK-grafy  
-mnozina grafov s rovnakym dK-rozdelenim ako vstupny graf  
-3K(g) ⊆ 2K(g) ⊆ 1K(g) ⊆ 0K(g)  
-nK – identicky graf

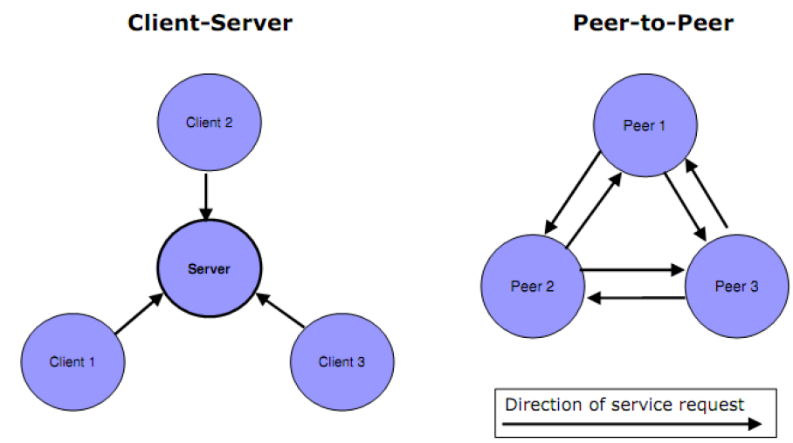
Zhrnutie vyvoja  
Obecne nahodne grafy => Pravdepodobnost hrany zavisla na vzdialenosti vrcholov => Zavedenie   
hierarchie => Power law => Principy rastov => dK-rozdelenie

Nahodny graf = mame graf s n vrcholmi G (n,p), neorientovana hrana vznikne medzi dvojicou vrcholov  
s pravdepodobnostou p, nezavisle na ostatnych dvojiciach uzlov.  
V modeli nahodneho grafu G(n,p): p = c (log n / n), ak c > 1 => vsetky grafy nemaju izolovane vrcholy,  
ak c < 1 => vsetky grafy maju aspon jeden izolovany vrchol.

Nahodny geometricky graf = G(n,r), dva uzly su spojene hranou, ak vzdialenost medzi nimi je <= r,  
vzdialenost medzi dvoma bodmi u =(x1,y1) a v (x2,y2) je: d(u,v) = max(|x1-x2|,|y1-y2|)

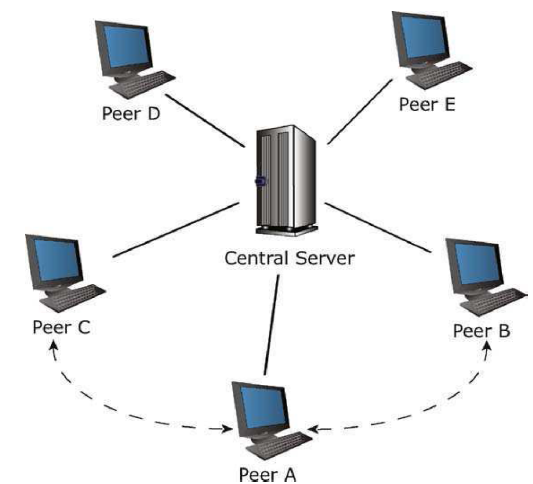
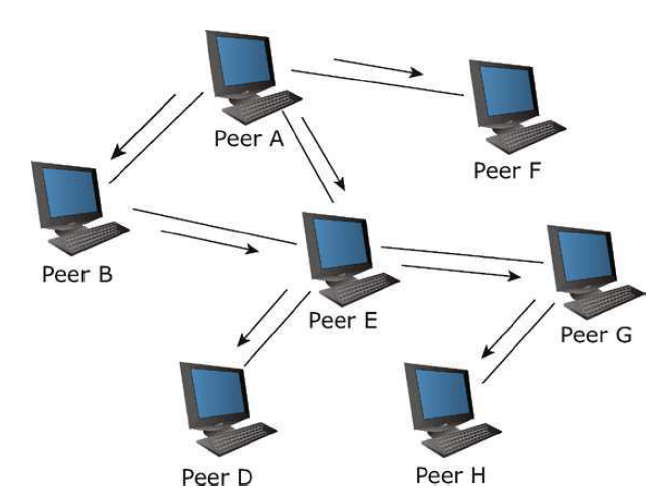
Pokial r >= sqrt ((c\*log n)/n), kde c je konstanta > 4, potom G(n,r) je asymptoticky suvisly graf  
=> Pravdepodobnost, ze G(n,r) je suvisly graf, sa blizi 1, ako sa n blizi nekonecnu

Prehladavanie v nestrukturovanych sietach  
1. Centralny index (Napster…)  
2. Decentralizovane (Gnutella…)



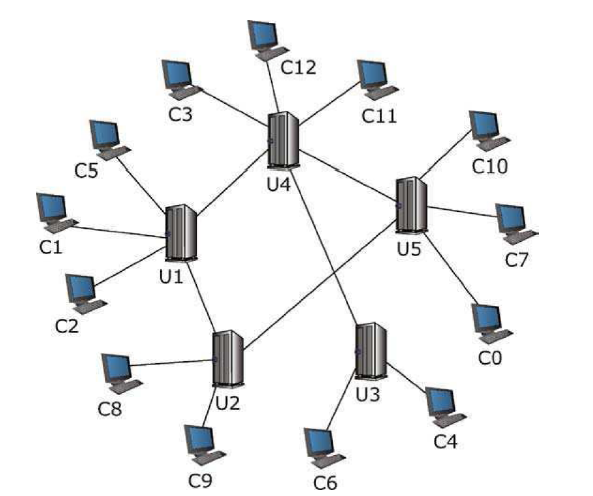
**P2P siete**

-“virtualna” siet utvorena nad existujucou sietovou infrastrukturou (napr. nad sietou Internet)  
-prekryvova siet sa vyuziva pre indexovanie a zistovanie susedov (peerov) => P2P system je   
 nezavisly na topologii fyzickej siete  
-novy peer musi za ucelom pripojenia k P2P sieti, ziskat informacie o najmenej jednom jej clenovi  
 -nevyhnutne sietove informacie: IP adresa, port, atd…



*Centralizovane P2P systemy Decentralizovane P2P systemy*

-Peer A zasiela dotaz na zoznam uzlov majucich -Peer A pozaduje urcite data, ktore su  
 pozadovane data centralnemu server, ulozene na peeroch D a H, poziadavka   
 ked uzol A ziska od centralneho server je broadcastovana vsetkym susedom  
 pozadovany zoznam, prevadza uz komunikaciu peeru A, potom vsetkym ich susedom atd…  
 sam napriamo s tymito uzlami az je doruceny vsetkym peerom v P2P sieti

Hybridne P2P systemy: Poziadavka na vyhladanie urcitej  
informacie je najprv smerovana na superpeer uzol  
nadradeny dotazujucemu sa uzlu, dany uzol v spolupraci  
s ostatnymi superpeerami vyhlada uzol, ktory  
pozadovane data spravuje a dotazujuci sa uzol na neho  
presmeruje

**Smerovanie v nestrukturovanych P2P sietach**

-kazdy peer spravuje svoje vlastne datove objekty a linky/spojenia ku svojim susedom  
-novy uzol kontaktuje nejaky uz pripojeny uzol a skopiruje si linky k jeho susedom  
-ziadny peer nema globalnu znalost o umiestneni datovych objektov v sieti  
-Techniky pre vyhladavanie v nestrukturovanych P2P sietach:  
 -Prehladavanie do sirky (Breadth-First Search, BFS) – napr. Gnutella  
 -Prehladavanie do hlbky (Depth-First Search, DFS) – napr. FreeNet  
 -Heuristicke smerovacie strategie

Heuristicke smerovacie strategie:  
1. Postupne vnorovanie (Iterative Deepening): kazdy dotaz je prevadzany sekvenciou niekolkych  
tradicnych BFS prehladavani s postupnym zvacsovanim prehladavacieho dosahu, proces vyhladavania  
je ukonceny bud najdenim vysledku alebo dosiahnutim maximalnej moznej hlbky prehladavania

2. Riadene BFS  
-kazdy z uzlov posiela dotaz len podmnozine svojich susedov  
-klucovou vlastnostou algoritmu je inteligentny vyber vhodnych susedov   
-vyhody: radikalne mensi pocet vyhladavacich sprav oproti standardnemu BFS  
-nevyhody: statistiky vedene voci susedom neobsahuju informacie suvisiace s obsahom dotazov

=> Inteligentne prehladavanie (Intelligent Search)  
-kazdy peer hodnoti svojich susedov na zaklade relevantnosti k danemu dotazu  
-dotaz, je poslany len najrelevantnejsim susedom  
-zaisteny preciznejsi vyber susedov ako u riadeneho BFS

3. Lokalne indexy (Local Indices Search)  
-kazdy uzol si vytvara indexy pre svoje lokalne data, aj pre data svojich susedov, ktori su od neho   
 vzdialeni k hopov (ak k=0 => metoda je podobna tradicnemu BFS, uzly spravuju indexy lokalnych dat)  
-navrateny vysledok rovnaky ako vysledok navrateny vsetkymi uzlami v rozsahu k hopov od uzlu  
-vyhody: redukcia vypocetnej narocnosti vdaka obmedzeniu spracovania dotazu na mensi pocet uzlov  
-nevyhody: 1. Vysoka ulozna kapacita, 2. Vysoka rezia na udrzbu indexov, 3. Zastaralost indexov

4. Nahodna prechadzka (Random Walk)  
-akonahle uzol prijme dotaz, preposle ho nahodne vybranemu susedovi ku spracovaniu  
-proces sa opakuje, kym nieje najdena odpoved alebo kym neexpiruje TTL (ak sa pouzije)

k-walker Random Walk Algorithm  
-povodca dotazu (zdrojovy uzol) zasiela k dotazovacich sprav svojim nahodne vybranym susedom  
 (miesto jednej spravy = 1-walker algorithm)  
-ked uzol prijme dotaz (tzv. walker), spracovava/preposiela ju s vyuzitim rovnakeho principu ako v   
 pripade zakladneho algoritmu nahodnej prechadzky (tj. jednemu vybranemu susedovi)  
-pocet sprav je navyseny linearne v porovnani s 1-walker algoritmom

Random Breadth First Search (RBFS)  
-podobne ako k-walker Random Walk  
-povodca dotazu (zdrojovy uzol) nahodne vyberie podmnozinu svojich susedov, ktorym posle dotaz  
-kazdy z tychto susedov znova nahodne vybera podmnozinu svojich susedov, ktorym posle dotaz, atd…  
-pocet sprav je navyseny exponencialne v porovnani s 1-walker algoritmom

5. Adaptivne pravdepodobnostne prehladavanie (Adaptive Probabilistic Search, APS)  
-kombinuje k-walker nahodnostne prehladavanie a pravdepodobnostne prehladavanie  
-zasiela dotaz susedom vybranym na zaklade urcitej pravdepodobnosti  
-na zaklade predoslych vysledkov, spravuje kazdy uzol pre kazdeho zo svojich susedov pravdepodobnost  
 jeho vybrania

Pristupy aktualizacie spravovanych pravdepodobnosti:

1.Optimisticky pristup – system proaktivne navysuje pravdepodobnost dotazovanym susedom a znizuje  
len vtedy, ked “walker” vyslany cez nich skonci s chybou

2.Pesimisticky pristup – system proaktivne znizuje pravdepodobnost dotazovanym susedom a zvysuje  
vtedy, ked “walker” vyslany cez nich skonci uspesnym najdenim vysledku

Swapping-APS: kazdy uzol priebezne prepina medzi optimistickym a pesimistickym pristupom  
Weighted-APS: berie v uvahu lokalitu susedov => blizsie uzly maju vyssiu pravdepodobnost

6. Interest-Based Shortcuts  
-kazdy uzol si spravuje doplnkove linky (interest-based short-cuts) pre vyssiu efektivitu prehladavania,  
 ktore spajaju dva uzly majuce podobne data  
-pri spracovani dotazu, je preposlany najprv uzlom na doplnkovych linkach, ak sa najde vysledok konci  
 prehladavanie, inak sa pouzije bezna prehladavacia metoda

**Smerovanie v strukturovanych sietach**

-v strukturovanych P2P sietach su uzly organizovane do urcitej topologie/urcitym usporiadanim  
-je garantovane, ze ak vysledok na dotaz v sieti existuje, bude najdeny  
-nevyhoda: nutnost udrzovania definovanej topologie vyzaduje vacsiu reziu v podobe udrziavania   
 smerovacich tabuliek

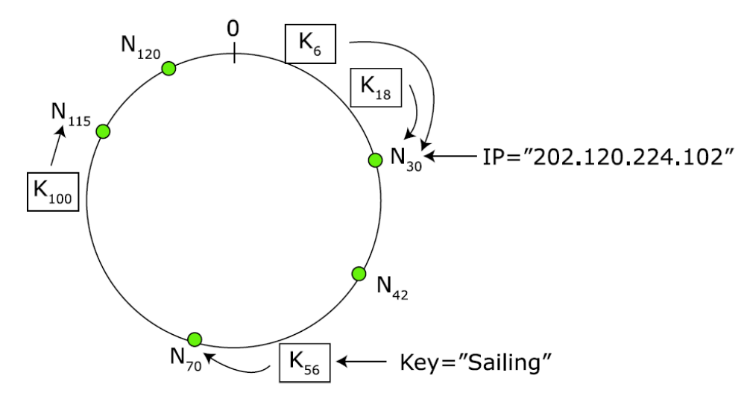
Podla vyuzitej prekryvnej topologiesa strukturovane P2P systemy delia:  
-Systemy vyuzivajuce distribuovane hashovacie tabulky – napr: Chord, CAN, Tapestry, atd…  
-Systemy vyuzivajuce tvz. Skip-listy – napr: Skip, Graph, SkipNet, atd…  
-Systemy vyuzivajuce stromove struktury – napr: P-Grid, P-Tree, BATON, atd…

Distributed hash tables, DHT (Distribuovane hash tabulky)  
-hash tabulky v p2p sietach, hladanie dat podla kluca  
-predpoklada sa ulozenie dat v distribuovanych uzloch (nie v poli)  
-pouziva sa prekryvova siet, ktora spaja uzly vhodnym sposobom  
-kazdy uzol p2p siete spravuje urcitu cast globalnej hashovacej tabulky  
-ulozenie/zistenie polozky s znamena oslovenie uzlu, ktory spravuje hodnotu hash(s)  
-riesenie pre ucinne hladanie aj riedko sa vyskytujucich objektov

Hash tabulky  
-rychle a lacne najdenie dat indexovanych podla kluca  
-obmedzeny sposob vyhladania dat (podla kluca) => strata obecnosti ale zisk efektivity a rychlosti

**Smerovacie strategie zalozene na DHT**

1.Chord  
-vyuziva jednosmernu hashovaciu funkciu pre mapovanie uzlov a datovych poloziek do jednosmerneho  
 priestoru m-bitovych identifikatorov  
-pre vygenerovanie identifikatoru uzlu je vyuzita IP adresa uzlu  
-pre vygenerovanie identifikatoru datovej polozky je vyuzita vlastna polozka (pripadne iny kluc)  
-priestor identifikatorov musi byt dost velky, aby pravdepodobnost priradenia rovnakeho identifikatoru  
 dvom roznym uzlom siete bola minimalna  
-priestor identifikatorov je logicka kruznica cisel v rozsrahu [0,2m-1]  
-system priraduje datovu polozku s klucom k ku prvemu uzlu n, ktoreho identifikator je rovny, pripadne  
 vyssi nez identifikator k datovej polozky (polozka s klucom k je priradena k prvemu uzlu v smere  
 hodinovych ruciciek od k)



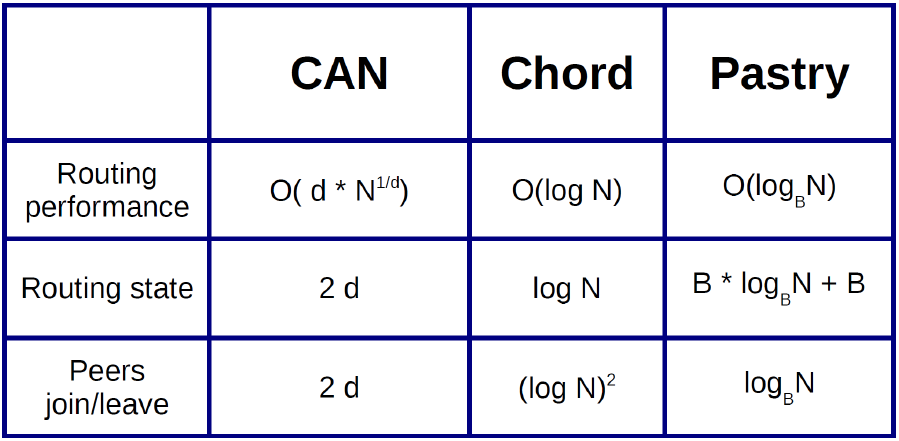
Jednoduchy vyhladavaci algoritmus:  
-kazdy uzol pozna len svojho bezprostredneho nasledovnika  
-ked uzol prijme poziadavku na vyhladanie:   
 -overi, ci on sam nema pozadovanu datovu polozku, ak ano => vyhladanie konci  
 -ak nie, posiela poziadavku svojmu bezprostrednemu nasledovnikovi  
 -vyhladanie konci: 1. Vysledok dotazu je najdeny  
 2. Identifikator bezprostredneho nasledovnika je vacsi ako identifikator  
 vyhladavanej datovej polozky => vysledok dotazu sa neda najst  
 -zlozitost vyhladavania je O(N) (N = pocet uzlov v systeme)

Skalovatelny vyhladavaci algoritmus:  
-kazdy uzol spravuje svoju vlastnu smerovaciu tabulku (finger table) obsahujucu zaznamy k jeho x  
 nasledovnikom  
-ked uzol prijme poziadavku na vyhladanie datovej polozky s identifikatorom k:  
 -ak dany uzol nespravuje hladanu datovu polozku, prehladava svoju smerovaciu tabulku  
 a hlada uzol n, pre ktory plati podmienka: n.id < n’.id < k  
 -ak taky uzol existuje, uzol n mu preposle poziadavku na vyhladanie datovej polozky  
 -ak taky uzol neexistuje, posle dotaz bezprostrednemu nasledovnikovi  
 -vyhladavanie konci: 1. Vysledok dotazu je najdeny  
 2. Identifikator bezprostredneho nasledovnika je vacsi ako identifikator  
 vyhladavanej datovej polozky => vysledok dotazu sa neda najst  
 -zlozitost vyhladavania: O(log N) (N = pocet uzlov v systeme)

2.Content Addressable Network (CAN)  
-smerovaci system utvoreny nad virtualnym d-dimenzionalnym kartezskym suradnicovym systemom  
-rozdeleny priestor datovych klucov do samostatnych oblasti, kazda oblast je spravovana jednym uzlom  
-tento uzol spravuje vsetky datove polozky vo svojej oblasti  
-vyuziva sa hashovacia funkcia pre mapovanie datovej polozky na bod p v suradnicovom systeme (d-tica)  
-kazdy uzol musi spravovat informacie o svojich susedoch (uzly spravujuce prilahle zony)  
-smerovanie: v kazdom kroku je hladany uzol, ktoreho suradnice su blizsie cielovej zone

3.Pastry  
-smerovaci system zalozeny na PRR stromoch  
-m-bitovy identifikator uzlu je rozdeleny na sekvenciu “cislic” so zakladom 2b  
 napr: 128-bitovy identifikator rozdeleny na 32 4-bitovych cislic => b=4, b=konfigurovatelny parameter  
-datova polozka je ulozena na uzol, s ktorym zdiela najdlhsi spolocny prefix identifikatora  
-kazdym krokom je vybrany uzol, ktory ma s cielovym uzlom dlhsi spolocny prefix  
-zlozitost hladania je O(log 2b N)   
-kazdy peer ma svoju smerovaciu tabulku (s pevnym poctom riadkov, v ramci kazdeho riadku obsahuje  
 rovnaky pocet poloziek (2b-1)  
-ID riadku = dlzka spolocneho prefix s hladanym uzlom  
-ID stlpca = dalsi mozny krok  
-kazdy uzol obsahuje aj zoznam tzv. leaf set  
-obsahuje zoznam ukazatelov na uzly, ktore su k danemu uzlu numericky najblizsie  
-sluzi ako zalozne riesenie ak sa v smerovacej tabulke nenajde uzol s dlhsim prefixom  
-velkost je typicky 2b   
-smerovanie v pastry:  
 -ak je hladany uzol v leaf set daneho uzlu, je smerovanie prevedene na zaklade tejto informacie  
 -inak sa ide podla smerovacej tabulky => posiela uzlu, ktory s cielovym zdiela dlhsi prefix  
 -ak je smerovacia tabulka prazdna alebo referencovany uzol je nedostupny:  
 -preposiela sa uzlu majucemu rovnako dlhy prefix, alebo  
 -uzlu, ktoreho identifikator je numericky blizsie hladanemu klucu

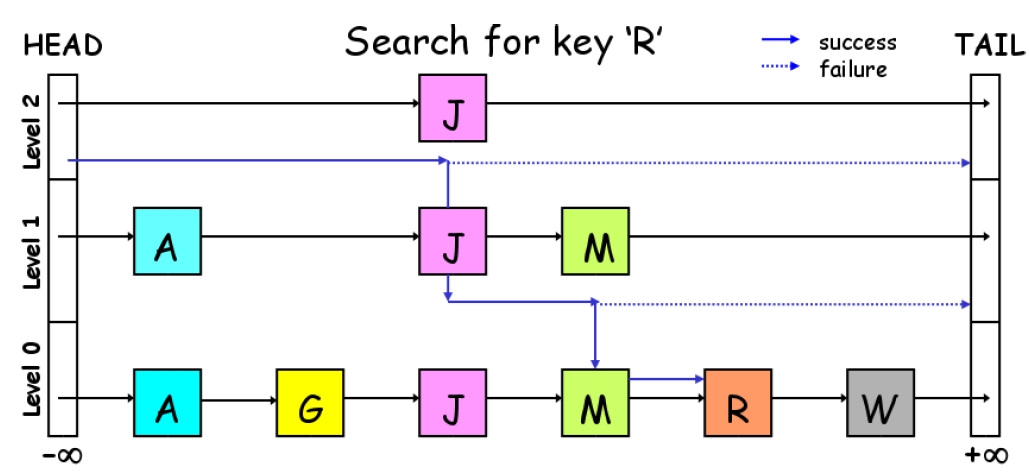
4.Tapestry  
-smerovacia strategia pre P2P siete zalozena na PRR stromoch, podobna Pastry  
-rozdiel oproti Pastry: v ramci Tapestry sa kazdym krokom pri hladani cieloveho uzla, hlada uzol,  
 s dlhsim spolocnym suffixom



**Smerovacie strategie zalozene na Skip Listoch**

Skip-List  
-datova struktura urcena pre ulozenie zotriedeneho zoznamu poloziek  
-vyuziva hierarchiu zretazenych zoznamov (jednoduchsie vyhladavanie/ukladanie)  
-zoznamy su zostavene do urovni: -najnizsia uroven (bezny zotriedeny zoznam)  
 -vyssie urovne, element z vrstvy I sa vyskytuje vo vrstve i+1  
 s urcitou pravdepodobnostou p (vacsinou p=1/2 alebo p=1/4)

Hladanie cieloveho elementu:  
-zacina na zaciatku zoznamu umiestneneho v najvyssej urovni a pokracuje horizontalne, pokial  
 nenajde polozku, ktora je vacsia alebo rovna hladanej  
-pokial je najdena polozka, ktora je rovna hladanej, je ciel najdeny  
-pokial je najdena polozka vacsia ako hladana, procedura sa vrati na predchadzajucu polozku  
 a zostupi o uroven nizsie, potom sa cela procedura opakuje  
-predpokladana cena hladania: (log1/pn)/p, p je konstanta => O(log n)



1.Skip Graph  
-smerovacia strategia zalozena na Skip-Listoch  
-narozdiel od originalnych skip-listov, poskytuje Skip Graph na kazdej urovni viac zoznamov  
-zoznamy, do ktorych dany uzol nalezi su urcovane podla nahodneho vektoru prislusnosti  
-pocet urovni je O(log N)

Vyhladavanie dotazu uzlom:  
-vyhladavanie zacina v najvyssej urovni, v zozname, do ktoreho dany uzol nalezi  
-v kazdom kroku: pokial na danej urovni existuje uzol, ktoreho hodnota je blizsie vyhladavanej,   
 je tomuto uzlu dotaz preposlany  
-inak dany uzol s vyhladavanim pokracuje na nizsej urovni  
-cielovy uzol je najdeny v najhorsom pripade na najnizsej urovni  
-zlozitost vyhladavania je O(log N)

2.SkipNet  
-smerovaci system podobny systemu Skip Graph  
-miesto skip-listov, su uzly organizovane do kruhov  
-kruhy su organizovane do urovni/levelov (ako u systemu Skip Graph)  
-uzly su na kazdej urovni zotriedene podla datoveho kluca  
-kazdy uzol ma na kazdej zo svojich urovni v smerovacej tabulke ulozene dva ukazatele k susedom  
 -ukazatele na urovni h ukazuju na uzly, ktore su cca 2h uzlov nalavo a napravo   
 -vsetky uzly su na najnizsej urovni(level 0) prepojene korenovym kruhom

**Stromovo orientovane smerovacie strategie**

1.P-Grid  
-zalozeny na virtualnom binarnom strome, v ramci ktoreho kazdy peer spravuje jeden list stromu  
-kazdy peer ma identifikator, ktory je zretazenim bitov reprezentujucich cestu od korena k uzlu  
-kazdy peer je zodpovedny za datove polozky, ktorych prefix identifikatoru kluca je zhodny  
 s identifikatorom daneho uzlu  
-k jednemu identifikatoru sa moze priradit viac peerov (zvysena odolnost voci vypadkom)  
-pre ucely vyhladavania kazdy peer spravuje vlastnu smerovaciu tabulku  
-neda sa zarucit vyvazenost stromovej struktury  
-zlozitost vyhladavania: O(log2 N)

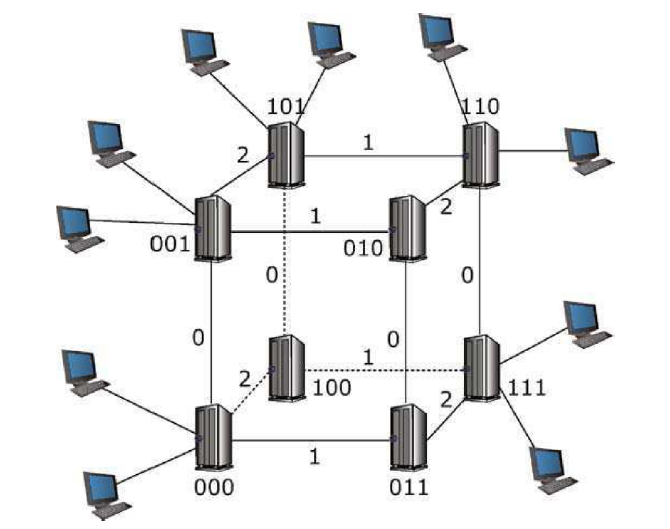
2.P-Tree  
-zalozeny na virtualnych vyvazenych B+ stromoch vybudovanych nad Chord strukturou  
-kazdy peer spravuje:   
1. uzol Chord siete, ktory je listom danej stromovej struktury  
2. Semi-independent B+ strom, co je peerov fully-independent B+ strom  
-fully-independent B+ strom peera, je B+ strom, v ktorom hodnota klucu ulozeneho na danom uzle  
 je povazovana za najmensiu v danej Chord strukture   
-semi-independent B+strom obsahuje len korenove uzly a vsetky uzly v najlavejsej casti daneho  
 fully-independent B+ stromu  
-pre ucely lahsej spravy sa mozu rozsahy uzlov danych B+ stromov prekryvat

3.BATON  
-data su ulozene na listoch stromov, aj na vnutornych uzloch  
-okrem liniek k rodicom a potomkom, su na kazdom uzle udrziavane linky k prilahlym uzlom a susedom  
-adjacent links (linky k prilahlym uzlom): vyuzite pre prepojenie uzlu s uzlom/uzlami, ktore spravuju  
 jemu prilahle rozsahy klucov  
-neighbour links (linky k susedom): vyuzite pre prepojenie uzlu so susedmi (rovnaka uroven stromu),   
 ktori su od neho vo vzdialenosti 2i, i>=0  
-ucelom tychto liniek, je zabranenie zahlteniu korena vyhladavacimi dotazmi

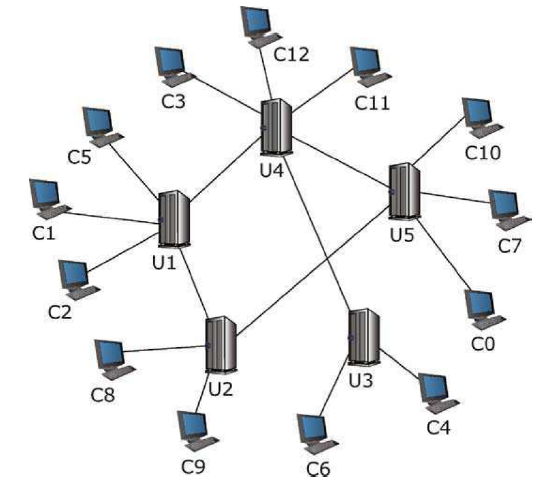
**Smerovanie v hybridnych P2P sietach**

Hybridne p2p systemy organizuju peery do hierarchickych struktur  
-vybrane uzly, su umiestnene do vyssej rovne  
-bezne peer uzly su umiestnene do nizsich urovni  
-kazdy klientsky uzol nalezi k jednemu superpeer uzlu a nekomunikuje s ostatnymi inak ako cez neho  
-priklady hybridnych sieti: KaZaA, BestPeer, Edutella, atd…

1.Edutella  
-dotaz na vyhladavanie je smerovany na superpeery, ktore su organizovane do HyperCup siete,  
 v ramci ktorej moze byt vyuzita smerovacia schema na bazi najdlhsieho spolocneho suffix



2.Ultrapeers (modifikovana siet Gnutella s ultrapeermi)  
-peer posle dotaz svojmu ultrapeerovi(superpeerovi), ten pomocou mechanizmu zaplavy rozosiela  
 dotaz do siete, ultrapeer, ktory spravuje peer s pozadovanymi datami, poskytne zdrojovemu peeru  
 IP adresu peeru, ktory ma pozadovane data



3.Structured Superpeers  
-superpeer uzly spravuju zoradene rozsahy klucov (0,4],(4,8],(8,12] a (12,0], pokial peer vyhladava  
 data s klucom K=10, najprv posiela dotaz na vyhladavanie svojmu superpeerovi, ten preposiela dotaz  
 superpeerovi, ktory spravuje data s rozsahom, do ktoreho spadaju hladane data, a tento superpeer   
 odpoveda zdrojovemu peeru s informaciou o IP adrese uzlu, ktory hladane data obsahuje

