

PB165 Grafy a sítě:  
Plánování pro transport, přenosy a komunikace

## 1 Transport

- Doba na nastavení
- Doba na dopravu

## 2 Plánování na počítačové síti

- Úvod k plánování úloh na počítačové síti
- Plánování datových přenosů
- Paralelní úlohy s komunikací

# Problém obchodního cestujícího

## Problém obchodního cestujícího

- obchodní cestující musí projet všechna města tak, aby
  - celková ujetá vzdálenost (resp. doba cesty) byla minimální a
  - každé město projel právě jednou

## Grafová reprezentace

- (orientovaný) hranově ohodnocený graf
- vrchol = město
- (orientovaná) hrana z A do B = přímá cesta z A do B
  - hrany mohou být orientované, pokud chceme uvažovat různou náročnost v opačných směrech cesty
- ohodnocení hrany z A do B = doba nutná na cestu z A do B

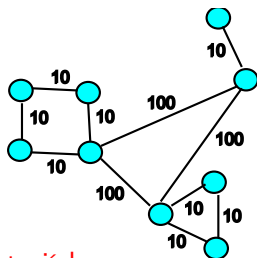
# Nastavovací doba a cena (*setup time and cost*)

- **Nastavovací doba**  $s_{ijk}, s_{jk}$ : **závislá na úlohách**
  - udává závislost na posloupnosti provádění úloh
  - $s_{ijk}$  čas nutný pro provádění úlohy  $k$  po úloze  $j$  na stroji  $i$
  - $s_{jk}$  nastavovací doba nezávislá na stroji
- **Problém obchodního cestujícího**  $= 1|s_{jk}|C_{\max}$ 
  - město reprezentuje úlohu s nulovou dobou provádění
  - $s_{jk}$ : čas na dopravu z města/úlohy  $k$  do města/úlohy  $j$
  - příklad: cesta přes města 12341 odpovídá  $s_{12} + s_{23} + s_{34} + s_{41}$
- **Klasický příklad: plnění limonád do lahví**  $1|s_{jk}|C_{\max}$ 
  - dána cena za nastavení stroje při změně plnění typu limonády
  - $s_{cola,voda} \quad s_{voda,cola} \quad s_{cola,dzus}$
  - posloupnost plnění: 100 lahví vody, 50 lahví coly, 70 lahví džusu,
- **Nastavovací cena**  $c_{ijk}, c_{jk}$ 
  - s přechodem lze spojit i cenu, kterou je nutno zaplatit

# Doba na dopravu (*transportation time*)

## Multi-operační rozvrhování

- úloha se skládá z několika operací
- může/nemusí být určeno pořadí operací
- operace má zadáno
  - dobu provádění, konkrétní stroj k provádění
- stroj: na každém stroji maximálně jedna operace
- doba na dopravu  $t_{hl}$  mezi stroji  $h$  a  $l$ : **závislá na strojích**
  - kapacita cest mezi stroji neomezená
  - délka cesty mezi stroji = součet odpovídajících dob na dopravu
- cíl: realizovat všechny operace všech úloh  
při minimalizaci času dokončení všech úloh



## Grafová reprezentace

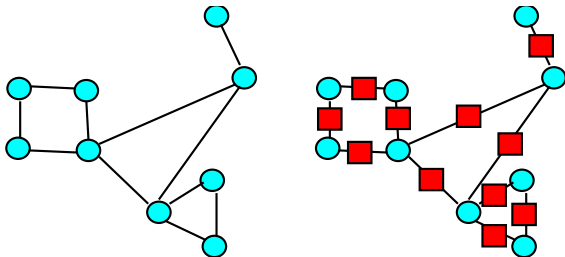
- **orientovaný hranově ohodnocený graf**
- vrchol: stroj
- hrana: pokud lze přejít přímo z jednoho stroje na druhý
- ohodnocení hrany: doba na dopravu z jednoho stroje na druhý

# Úvod k plánování úloh na počítačové síti

- Stroj: dán počet procesorů
- Úlohy prováděny na jednom nebo více uzlech počítačové sítě
  - vyžadují několik procesorů
- Úlohy potřebují k výpočtu **data**
  - data dané velikosti na jednom nebo více uzlech
  - data je nutné přenést na uzel, kde se úloha bude počítat
  - realita: data jsou často zreplikována na několika uzlech
- Linka:
  - **kapacita linky**: velikost přenesených dat za časovou jednotku
    - např. 100Mb/s, 1Gb/s, 10Gb/s
  - **latence**: doba nutná na přenos dat po lince
- Cíl: **realizovat všechny úlohy tak, jak dynamicky přibývají**
  - úlohy musí mít dostatek procesorů
  - data musí ležet v době výpočtu na uzlu, kde se počítá úloha
  - je nutné plánovat i přenosy dat tak, aby bylo možné data přenést vzhledem k latenci i kapacitě linek na cestě

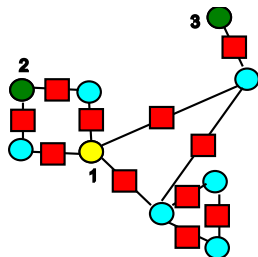
# Počítačová síť: grafová reprezentace

- Vrcholově ohodnocený neorientovaný graf
- Vrchol: stroj nebo linka
- Ohodnocení vrcholu-stroje: počet procesorů
- Ohodnocení vrcholu-linky: kapacita linky
  - linka je chápána jako zdroj, který má zadánu kapacitu
  - doba zpracování úlohy na lince (tj. přenosu dat pro úlohu na lince) odpovídá latenci
- Hrany: pokud jsou stroje A a B přímo spojeny linkou C, pak existují hrany AC a BC

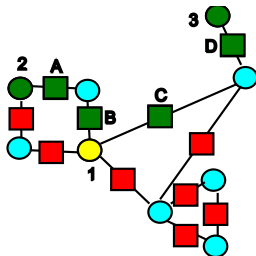


# Plánování úlohy na počítačové síti: příklad

- Úloha naplánována k provádění na uzlu 1
- Data na uzlech 2 a 3



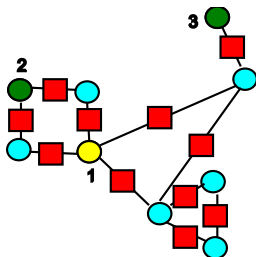
- Data jsou přenesena přes D,C a A,B
- Kapacita linek A,B,C,D musí být v daném čase postačující
- Celková doba přenosu do 1:  
 $\max(\text{latenceA} + \text{latenceB}, \text{latenceD} + \text{latenceC})$



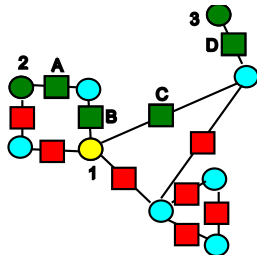


# Plánování úlohy na počítačové síti: příklad

- Úloha naplánována k provádění na uzlu 1
- Data na uzlech 2 a 3



- Data jsou přenesena přes D,C a A,B
- Kapacita linek A,B,C,D musí být v daném čase postačující
- Celková doba přenosu do 1:  $\max(\text{latenceA} + \text{latenceB}, \text{latenceD} + \text{latenceC})$



## Otázky:

- Je možné takovouto úlohu naplánovat za probíhajícího provozu na síti?
- Je možné ji naplánovat při modifikaci cest pro přenosy?
- Obecně: jak naplánovat úlohu(y) za daného provozu na síť?

směrování

## Problém:

- plánování datových přenosů, kde se velikost přenášených dat blíží kapacitám používaných linek

## Varianty:

- plánování přenosů v čase
  - je třeba uvažovat plánování zdrojů v čase
- plánování současně probíhajících přenosů
  - není uvažován čas, vše se plánuje pro aktuální okamžik
  - není tedy ani uvažováno plánování úloh v čase na uzly
  - pro tento případ uvedeme základní model

# Příklady aplikací plánování přenosů

## Plánování datových přenosů pro speciální zařízení a přístroje

- umožňuje plánovat přenosy dopředu na danou dobu, kdy budou linky dostupné (bulk přenosy dat)
- např. RHIC (relativistic heavy ion collider, USA)

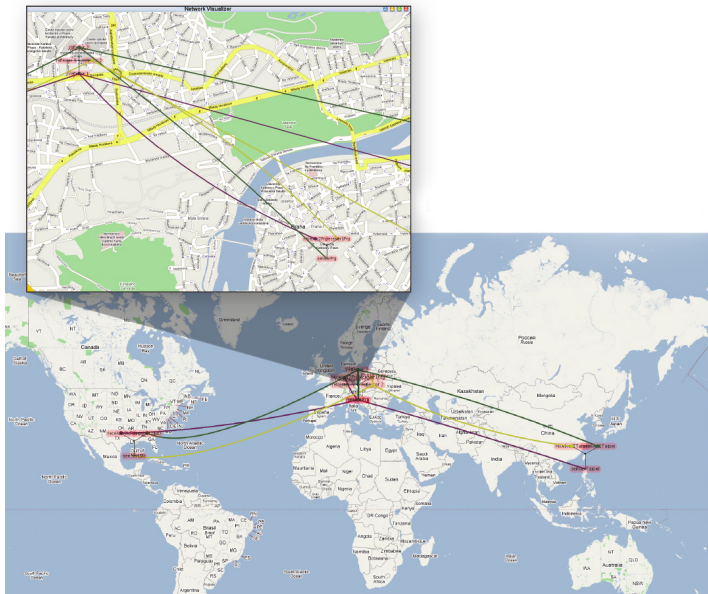
## Umístění aplikací na servery

- aplikace je nutno naplánovat na serverech, plánování v daném okamžiku
- ceny: za přenos aplikací na servery, za běh aplikace na daném serveru
- např. aplikace pro italskou mezibankovní síť

## Plánování přenosů HD videa v rámci kolaborativního prostředí

- nutná reakce na okamžitý stav počítačové sítě a naplánování na síť za daných aktuálních podmínek
- např. medicínské aplikace, filmový průmysl, výuka pomocí videa (vzdálená výuka, přístup např. k operačním sálům na medicíně, záznam a zpracování přednášek)
- systém CoUniverse vyvíjený na FI MU
  - <https://www.sitola.cz/CoUniverse>

# Naplánované spojení pomocí CoUniverse



Na **uzlu**  $v \in V$  se mohou nacházet různé aplikace

- **producent**  $p \in P$  produkuje data
- **konzument**  $c \in C$  přijímá data
- **distributor**  $d \in D$  je schopen rozesílat přijímaná data do několika linek distributor jako náročná aplikace nejvýše jeden na uzlu  
možné varianty distributora:
  - 1 distributor může data transformovat (transkódovat) přenášená data
    - např. z nekomprimovaného HDTV videa na HDV MPEG2 video
  - 2 data jsou stejného typu, tj. distributor je reflektorem (uvažováno dále)

Uzel má několik rozhraní (interfaces), po kterých posílá a přijímá data

- množina všech **rozhraní**  $I$
- kapacita rozhraní  $capacity_I(i)$  pro  $i \in I$
- linky připojené k rozhraní  $links(i)$  pro  $i \in I$

# Grafová reprezentace (dokončení)

Linka  $l \in L$  má určen

- počáteční uzel  $begin(l)$
- koncový uzel  $end(l)$
- latenci  $latency(l)$
- kapacitu  $capacity(l)$

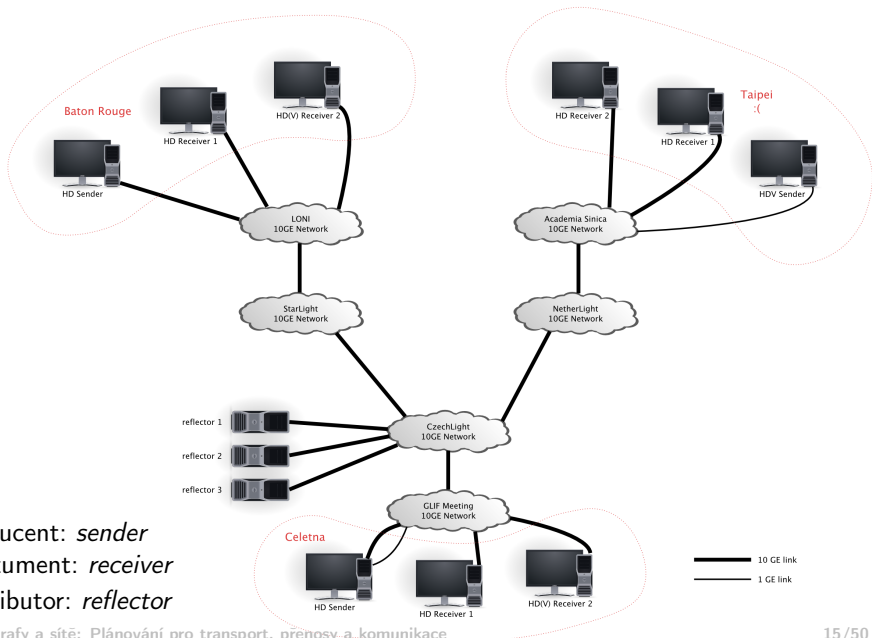
Linky reprezentují logickou strukturu sítě

- reálná struktura sítě často neznámá
- zahlcení linky detekováno monitorováním stavu sítě
- v případě nutnosti možné přepínání
  - nutné realizovat v reálném čase  
tj. vyžaduje rychlou odezvu

Orientovaný graf  $(V, L)$

- vrcholy: množina uzlů  $v \in V$
- hrany: množina orientovaných linek  $l \in L$
- tento graf reprezentuje základní strukturu, se kterou pracujeme

# Ukázka sítě při plánování datových přenosů



Producent: *sender*  
Konzument: *receiver*  
Distributor: *reflector*

**Stream**  $s \in S$  je datový tok od producenta ke konzumentům

- $bandwidth(s)$  udává požadovanou **šířku pásma** datového toku streamu
- příjem dat streamu  $s$  je požadován množinou zadaných konzumentů  $consumers(s) \subseteq C$
- data streamu  $s$  vysílá právě jeden producent  $p = producer(s) \in P$ 
  - *obecně* může existovat více producentů a cílem plánování je pak vybrat producenty zajišťující maximální kvalitu danou typem přenášených dat
    - např. nekomprimované HDTV video, HDV MPEG2 video, HDV MPEG4 video
  - pro *zjednodušení* můžeme předpokládat, že každý stream má předem jednoznačně určeného producenta
    - vhodného producenta lze vypočítat předem

Předpokládáme tedy: **pro každý stream přesně určen jeden producent**



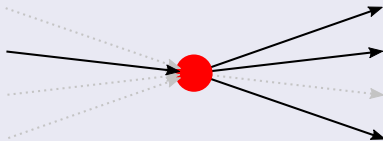
## Problém plánování několika streamu

- naplánovat všechny streamy z  $S$  na dostupné linky tak, aby byla optimalizována zadaná kritéria, např.
  - minimalizace celkové latence
  - při uvažování kvality spojení: maximalizace kvality přenášených dat
- všechny přenosy probíhají zároveň

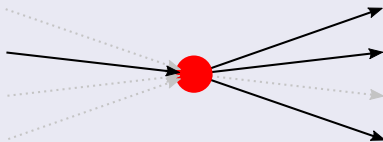
## Konstrukce distribučního stromu pro každého producenta/stream $s$

- distribuční strom jako podgraf grafu  $(V, L)$
- kořen stromu: uzel s producentem  $producer(s)$
- listy stromu: uzly s konzumenty  $consumers(s)$
- vnitřní uzly stromu: uzly s distributory

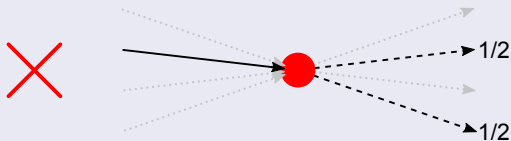
## Distributor vytváří kopie dat



## Distributor vytváří kopie dat



## Všechny pakety jdou k jednomu konzumentovi po jedné cestě



## Modely

- model založený na linkách
- model založený na cestách
- model založený na uzlech

### Model založený na linkách (*link-based model*)

- pro každý požadavek (stream)  $s$ : proměnná pro každý link  $l$
- **Proměnná**  $x_{s,l}$  pro každý stream  $s$  a každou linku  $l$

$$x_{s,l} = \begin{cases} 1 & \text{jestliže } s \text{ prochází po } l \\ 0 & \text{jinak} \end{cases}$$

- **Cíl:** nalézt hodnoty proměnných  $x_{s,l}$  optimální vzhledem k zadanému optimalizačnímu kritériu
- pro tento model si ukážeme vyjádření typických požadavků problému

## Model založený na cestách (*path-based model*)

- pro každý požadavek  $s$ : proměnná pro každou možnou cestu  $p$  mezi zdrojem (producent) a cílem (konzument)

$$x_{s,p} = \begin{cases} 1 & \text{jestliže jde požadavek } s \text{ po cestě } p \\ 0 & \text{jinak} \end{cases}$$

- náročné pokud je hodně možných cest

## Model založený na uzlech (*node-based model*)

- pro každý požadavek  $s$ : proměnná pro každý uzel  $v$

$$x_{s,v} = \begin{cases} 1 & \text{jestliže je uzel } v \text{ použit pro požadavek } s \\ 0 & \text{jinak} \end{cases}$$

## Konzistentní plánování

- kapacita linek a rozhraní
- posílání a přijímání dat
- vlastnosti konzumenta a producenta
- vlastnosti distributora
- eliminace cyklů

## Optimální plánování

- optimalizační kritéria

**Celková latence** musí být minimalizována

$$\text{minimize } \sum_{s \in S} \sum_{l \in L} \text{latency}(l) \cdot x_{s,l}$$

Další kritéria:

- **maximalizace kvality přenosu**
  - při zajišťování linek s maximální kapacitou pro přenášená data
- **balancování latencí všech linek**
  - při video-konferencích k zajištění bezproblémového přerušení a vstupu do probíhající video-konference

## Kapacita linky

- Požadovaná šířka pásma pro všechny streamy nesmí překročit kapacitu žádného linku

$$\forall l \in L \quad \sum_{s \in S} \text{bandwidth}(s) \cdot x_{s,l} \leq \text{capacity}(l)$$



## Kapacita linky

- Požadovaná šířka pásma pro všechny streamy nesmí překročit kapacitu žádného linku

$$\forall l \in L \quad \sum_{s \in S} \text{bandwidth}(s) \cdot x_{s,l} \leq \text{capacity}(l)$$

## Kapacita rozhraní

- Streamy přenášené jedním rozhraním nesmí překročit jeho kapacitu

$$\forall i \in I \quad \sum_{s \in S} \sum_{l \in \text{links}(i)} \text{bandwidth}(s) \cdot x_{s,l} \leq \text{capacity}_l(i)$$

## Kdo může posílat data?

- Pokud není na počátečním uzlu linky  $l$   
ani producent streamu  $s$  ani distributor  $\Rightarrow$

$$x_{s,l} = 0$$

## Kdo může posílat data?

- Pokud není na počátečním uzlu linky  $l$   
ani producent streamu  $s$  ani distributor  $\Rightarrow$

$$x_{s,l} = 0$$

## Kdo může přijímat data?

- Pokud není na koncovém uzlu linky  $l$   
ani konzument streamu  $s$  ani distributor  $\Rightarrow$

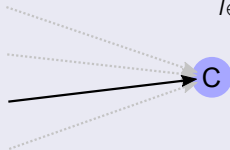
$$x_{s,l} = 0$$

# Vlastnosti konzumenta a producenta

Konzument  $c$  přijímá data právě jednou linkou

$$\forall s \in S \forall c \in \text{consumers}(s)$$

$$\sum_{l \in L \wedge c \in \text{end}(l)} x_{s,l} = 1$$

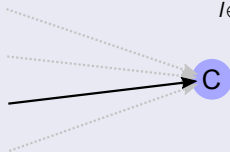


# Vlastnosti konzumenta a producenta

## Konzument $c$ přijímá data právě jednou linkou

$$\forall s \in S \forall c \in \text{consumers}(s)$$

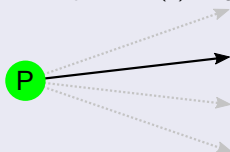
$$\sum_{l \in L \wedge c \in \text{end}(l)} x_{s,l} = 1$$



## Producent posílá data právě jednou linkou

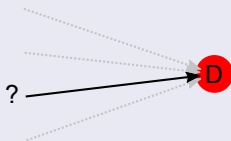
$$\forall s \in S$$

$$\sum_{l \in L \wedge \text{producer}(s) \in \text{begin}(l)} x_{s,l} = 1$$



## Distributor přijímá data nejvýše jednou

$$\forall d \in D \quad \sum_{s \in S} \sum_{l \in \text{inlinks}(d)} x_{s,l} \leq 1$$

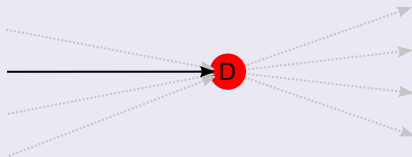


- $\text{inlinks}(d)$  linky, které vstupují do uzlu distributora
- $\text{outlinks}(d)$  linky, které vystupují z uzlu distributora

## Distributor přeposílá příchozí stream

- **aktivní distributor** pro stream  $s$

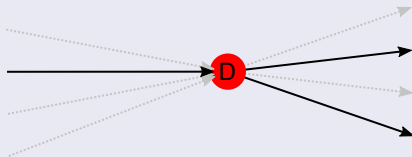
$$\forall d \in D \quad \sum_{l \in \text{inlinks}(d)} x_{s,l} \leq \sum_{l \in \text{outlinks}(d)} x_{s,l}$$



## Distributor přeposílá příchozí stream

- **aktivní distributor** pro stream  $s$

$$\forall d \in D \quad \sum_{l \in \text{inlinks}(d)} x_{s,l} \leq \sum_{l \in \text{outlinks}(d)} x_{s,l}$$

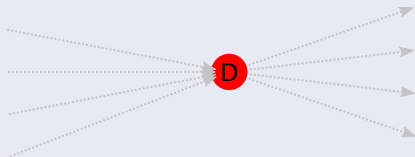


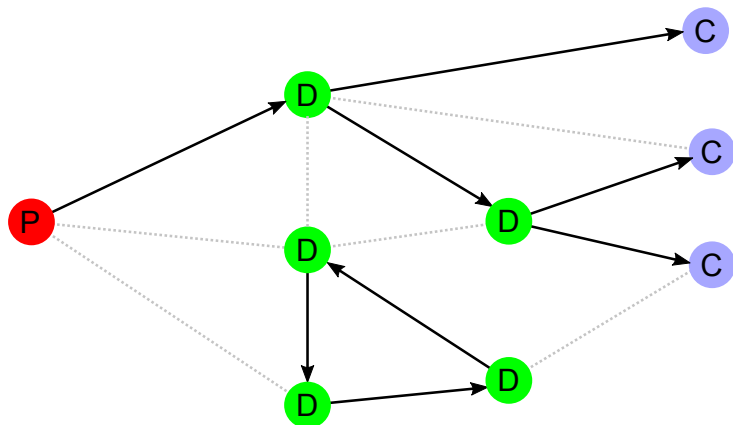


## Na distributorovi nevznikají žádná data

- **neaktivní distributor** pro stream  $s$

$$\forall d \in D \quad \sum_{l \in \text{inlinks}(d)} x_{s,l} = 0 \Leftrightarrow \sum_{l \in \text{outlinks}(d)} x_{s,l} = 0$$

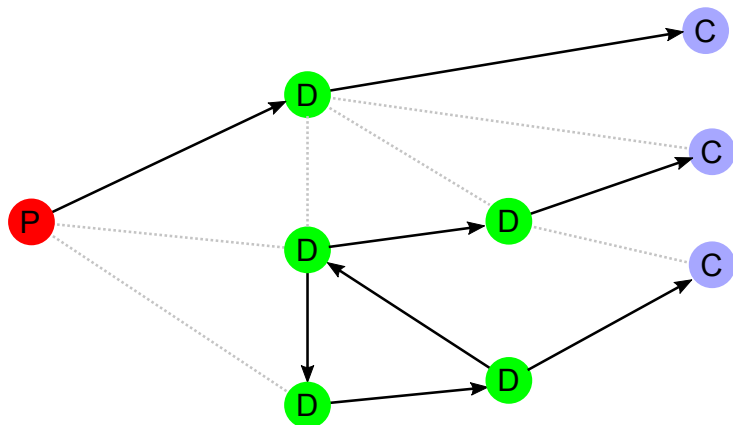




- Všechny cykly mezi distributory je nutné zakázat

Řešení:

- Nejvýše  $k - 1$  používaných linek mezi každou  $k$ -ticí uzlů pro každý stream
  - ve stromu s  $k$  uzly je  $k - 1$  hran, přidáním další hrany vznikne cyklus



- Všechny cykly mezi distributory je nutné zakázat

Řešení:

- Nejvýše  $k - 1$  používaných linek mezi každou  $k$ -ticí uzlů pro každý stream
  - ve stromu s  $k$  uzly je  $k - 1$  hran, přidáním další hrany vznikne cyklus

- Uvedený model popisuje základní množinu požadavků pro řešení problémů plánování datových přenosů
- Při řešení konkrétního problému nutné přidat specifická omezení a také redundantní omezení pro úspěšné/efektivní řešení problému
- Daný model (a jeho rozšíření na konkrétní aplikaci) lze řešit pomocí
  - grafových algoritmů (toky v síti, hledání cest v grafu)
  - programování s omezujícími podmínkami
    - např. CoUniverse využívá pro plánování Java řešič Choco pro omezující podmínky
  - celočíselné programování
    - nově vyvíjený plánovač pro CoUniverse
  - meta-heuristiky (lokální prohledávání)

# Datové přenosy: příklad

Pro zadanou počítačovou síť uveďte seznam proměnných. Navrhněte, jak by mohl být datový přednos realizován a uveďte odpovídající hodnoty proměnných reprezentující řešení optimalizující celkovou latenci. Jaká je celková latence?

Ohodnocení linku  $l$

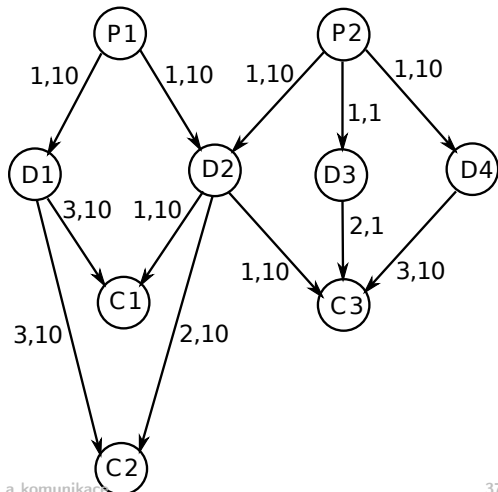
- $latency(l), capacity(l)$

Stream  $s_1$

- $bandwidth(s_1) = 5$
- $producer(s_1) = P_1$
- $consumers(s_1) = \{C_1, C_2\}$

Stream  $s_2$

- $bandwidth(s_2) = 5$
- $producer(s_2) = P_2$
- $consumers(s_2) = \{C_3\}$



# Datové přednosy: výsledný graf

Proměnné:

• pro stream  $s_1$ :  $x_{s_1,1}, \dots, x_{s_1,12}$

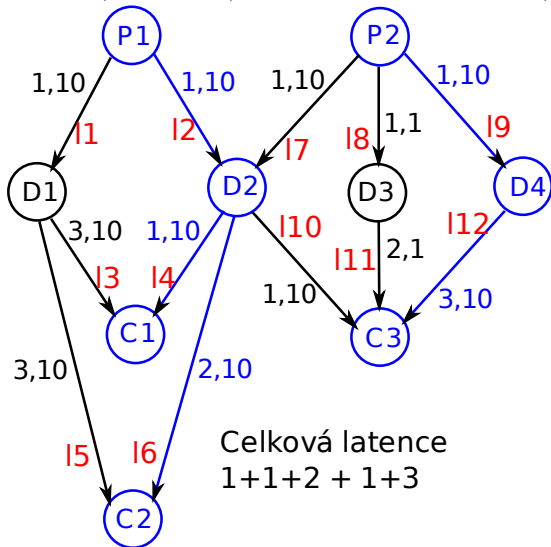
pro stream  $s_2$ :  $x_{s_2,1}, \dots, x_{s_2,12}$

$$x_{s_1,2} = 1$$

$$x_{s_1,4} = 1$$

$$x_{s_1,6} = 1$$

pro ostatní linky  
a stream  $s_1$  ... 0



$$x_{s_2,9} = 1$$

$$x_{s_2,12} = 1$$

pro ostatní linky  
a stream  $s_2$  ... 0

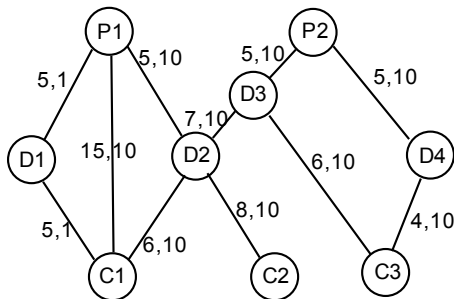
Celková latence  
 $1+1+2+1+3$

# Datové přenosy: cvičení

Pro graf počítačové sítě na obrázku uvažujte datové přenosy pro stream  $s_1$  a  $s_2$ , které jsou zadány následujícím způsobem

- stream  $s_1$ :  $\text{bandwith}(s_1)=10$ ,  $\text{producer}(s_1)=P1$ ,  $\text{consumer}(s_1)=\{C1\}$ ;
- stream  $s_2$ :  $\text{bandwith}(s_2)=10$ ,  $\text{producer}(s_2)=P2$ ,  $\text{consumer}(s_2)=\{C2,C3\}$ .

Ohodnocení hran grafu přitom udává jejich latenci a kapacitu.

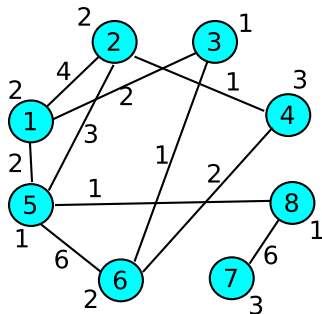


Uveďte, jaké proměnné jsou nutné při řešení tohoto *problému datových přenosů* a navrhnete jejich vhodné hodnoty tak, aby odpovídající řešení optimalizovalo celkovou latenci. Jaká je pak celková latence?

# Paralelní komunikující úlohy

## Paralelní aplikace

- $n$  komunikujících úloh
- $m$  procesorů
- několik úloh prováděno **zároveň** na každém procesoru
- tj. opět plánování pro daný časový okamžik
- kromě přenosů explicitně uvažována zátěž na uzlech



## Grafová reprezentace

- **hranově a vrcholově ohodnocený neorientovaný graf**
- ohodnocené vrcholy: úlohy s danou výpočetní náročností
- ohodnocené hrany: **průběžně** komunikující úlohy s komunikační náročností



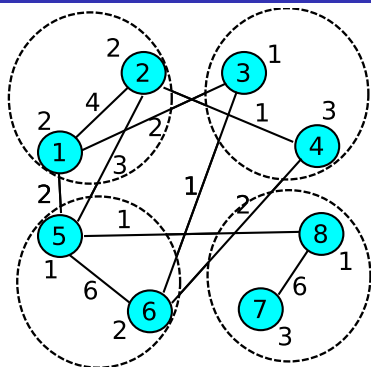
# Paralelní komunikující úlohy

## Paralelní aplikace

- $n$  komunikujících úloh
- $m$  procesorů
- několik úloh prováděno **zároveň** na každém procesoru
- tj. opět plánování pro daný časový okamžik
- kromě přenosů explicitně uvažována zátěž na uzlech

## Grafová reprezentace

- **hranově a vrcholově ohodnocený neorientovaný graf**
- ohodnocené vrcholy: úlohy s danou výpočetní náročností
- ohodnocené hrany: **průběžně** komunikující úlohy s komunikační náročností



## Vyvažování zátěže (*load balancing*):

přirazení úloh na procesory tak, aby

- byla vyvážená zátěž jednotlivých procesorů
- byla minimalizována komunikace úloh na různých procesorech

Formulace problému vyvažování zátěže jako  
problému rozdělení grafu (*graph partitioning*)

Rozdělení grafu  $G = (V, E)$  na  $V = V_1 \cup \dots \cup V_m$  tak, že je

- $V_1 \cap \dots \cap V_m = \emptyset$
- $G_1 = (V_1, E_1), \dots, G_m = (V_m, E_m)$
- $E_i$  tvořeno hranami, jejichž oba vrcholy patří do  $V_i$
- součet ohodnocení vrcholů v jednotlivých  $V_i$  „zhruba stejný“
- součet ohodnocení hran  $E \setminus \{E_1 \cup \dots \cup E_m\}$  spojujících různé  $V_j$  a  $V_k$  minimalizován

Speciální případ:  $V = V_1 \cup V_2$  bisekce grafu (graph bisection), tj. z grafu  $G = (V, E)$  vytvoříme dva podgrafy  $(V_1, E_1)$   $(V_2, E_2)$  tak, že

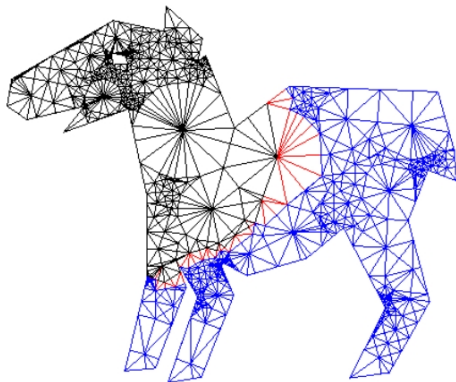
- $V = V_1 \cup V_2, V_1 \cap V_2 = \emptyset$
- $E_i$  tvořeno hranami, jejichž oba vrcholy patří do  $V_i$ , tj.  
 $E_1, E_2 \subset E, E_1 \cap E_2 = \emptyset, E_i$
- součet ohodnocení vrcholů ve  $V_1$  a  $V_2$  je „zhruba stejný“
- součet ohodnocení hran  $E \setminus \{E_1 \cup E_2\}$  spojující vrcholy z  $V_1$  a  $V_2$  je minimalizován

Jak nalézt vhodné rozdělení grafu?

- problém optimálního rozdělení je NP-úplný
  - už pro bisekci: prohledání všech podmnožin množiny vrcholů (podmnožina a její doplněk tvoří  $V_1$  a  $V_2$ )
- nutné použít dobré heuristiky

# Heuristika: opakovaná bisekce grafu

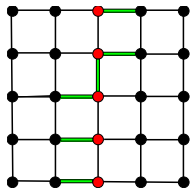
Základní používaný princip při rozdělení grafu:  
rozdělení množiny vrcholů  $V$  na  $2^k$  částí:  
rekurzivní bisekce grafu  $k$ -krát



58 dělicích hran

# Dělicí hrany (edge separator) vs. dělicí vrcholy (vertex separator)

- **Dělicí hrany:**  $E_S \subseteq E$  dělí  $G$  po odstranění  $E_S$  z  $E$  na dvě stejně velké nesouvisející komponenty  $V$ :  $V_1$  a  $V_2$
- **Dělicí vrcholy:**  $V_S \subseteq V$  dělí  $G$  po odstranění  $V_S$  a všech jejich hran na dvě stejně velké nesouvisející komponenty  $V$ :  $V_1$  a  $V_2$



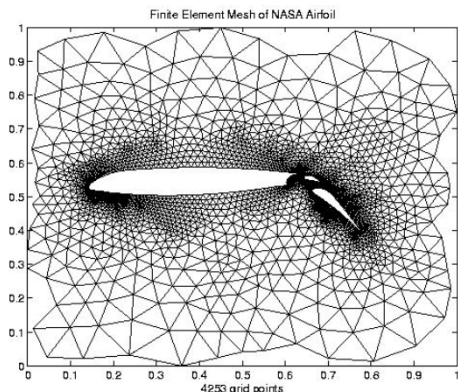
$E_S =$  zelené hrany

$V_S =$  červené vrcholy

# Rozdělení se souřadnicemi vrcholů (partitioning with nodal coordinates)

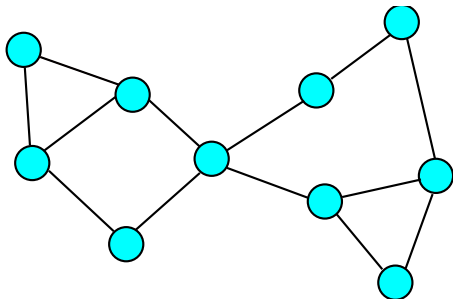
Myšlenka rozdělení pomocí souřadnic vrcholů

- každý vrchol má souřadnice v prostoru → **rozdělení prostoru**

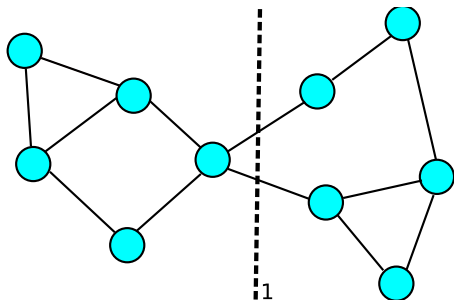


- pomocí dělicí přímky, která dělí vrcholy v prostoru na poloviny

# Opakovaná bisekce grafu s dělicí přímkou: příklad

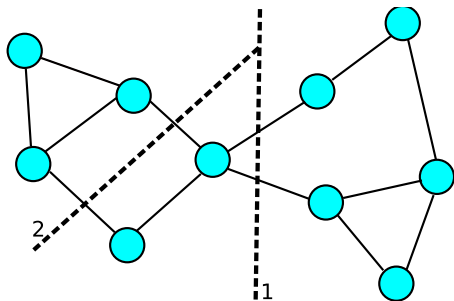


# Opakovaná bisekce grafu s dělicí přímkou: příklad





# Opakovaná bisekce grafu s dělicí přímkou: příklad



# Opakovaná bisekce grafu s dělicí přímkou: příklad

