On-line rozvrhování periodických úloh pod RT OS

I při využití jádra plánovače je potřeba **zadat prioritu procesů**

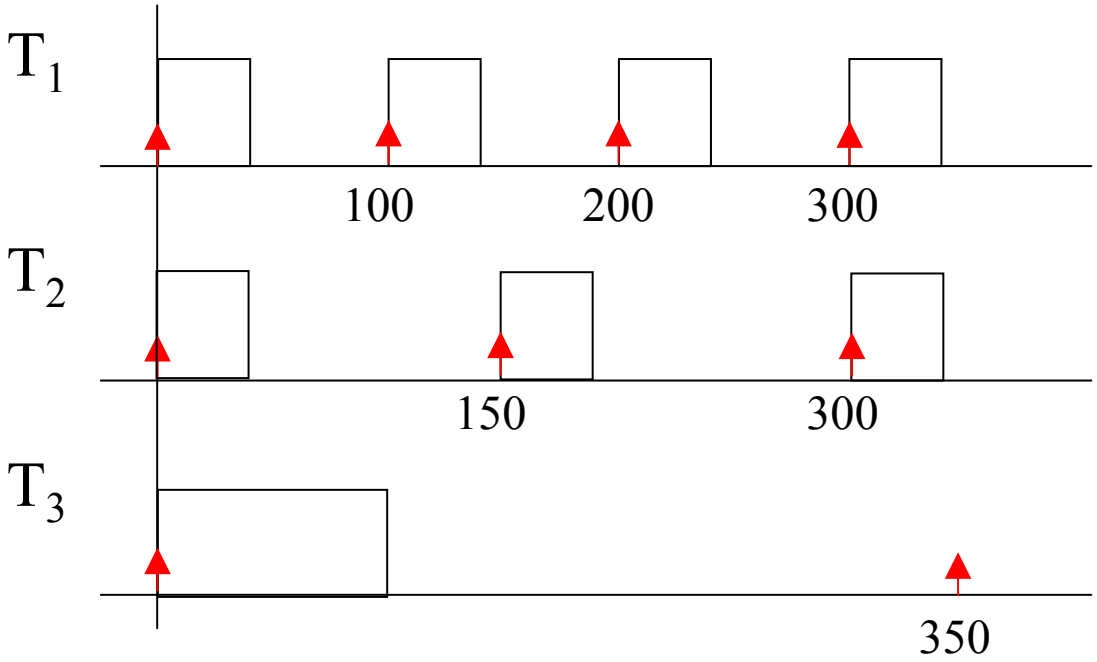
- problém rozvržení aplikace je řešen:
  - statické přiřazením priorit $\Rightarrow$ RMS
  - on-line přidělování procesoru $\Rightarrow$ jádro OS

© Z. Hanzálek 2005

Typická RT aplikace

- RT ⇒ deadline
- periodické rozvrhování (neperiodické úlohy se rozvrhují pomocí tzv. serverů)

Příklad 1:



Úloha T_1:

processing time…..p_1=40;

period……..τ_1=100;

Úloha T_2:

p_2=40;

τ_2=150;

Úloha T_3:

p_3=100;

τ_3=350;

© Z. Hanzálek 2005

Rate Monotonic Scheduling (RMS)

Podmínky:

- úlohy vykonávané preemptivním jádrem
- deadline úlohy je na konci každé periody

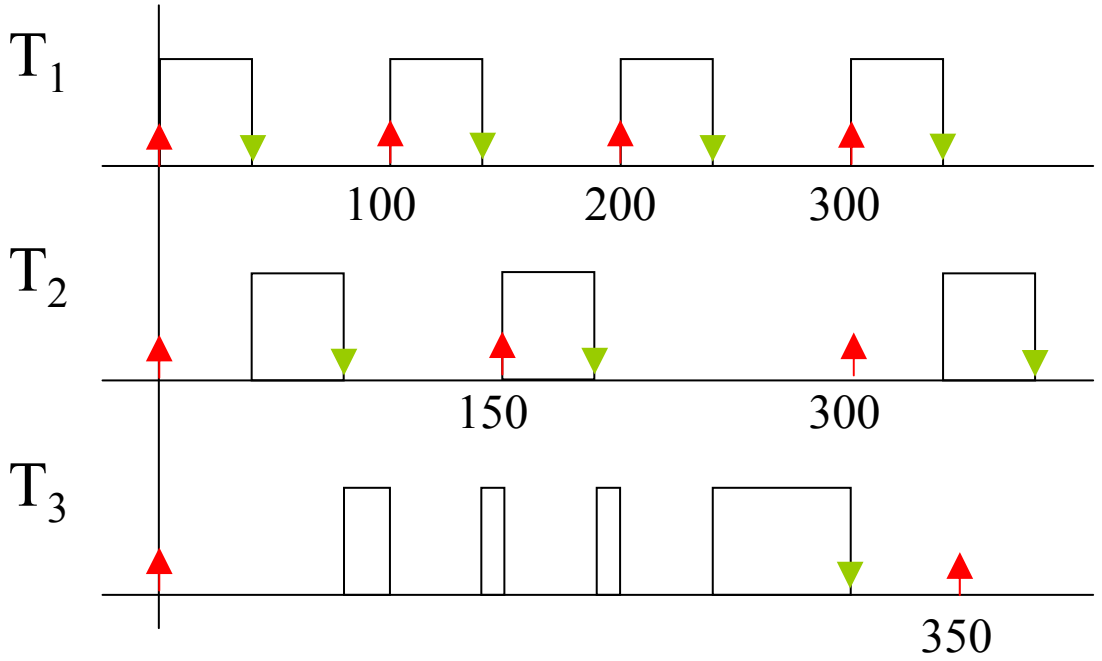
Algoritmus: přiřaď statické priority **podle jejich periody (tj. deadline).** Vyšší priority úlohám s větší frekvencí.

Záruka rozvrhovatelenosti: ***n*** **periodických** a **nezávislých** úloh bylo vykonáváno před deadline

Optimalita: neexistuje algoritmus se **statickým** přidělováním priorit, který by rozvrhl aplikaci, jež nelze rozvrhnout pomocí RMS

© Z. Hanzálek 2005

Řešení příkladu 1 pomocí RMS



konec periody (zároveň deadline)

ukončení úlohy (completion time)

© Z. Hanzálek 2005

Zatížení procesoru

Zatížení procesoru $\boldsymbol{U}$ (processor utilization factor) vyjadřuje efektivitu jeho využití.

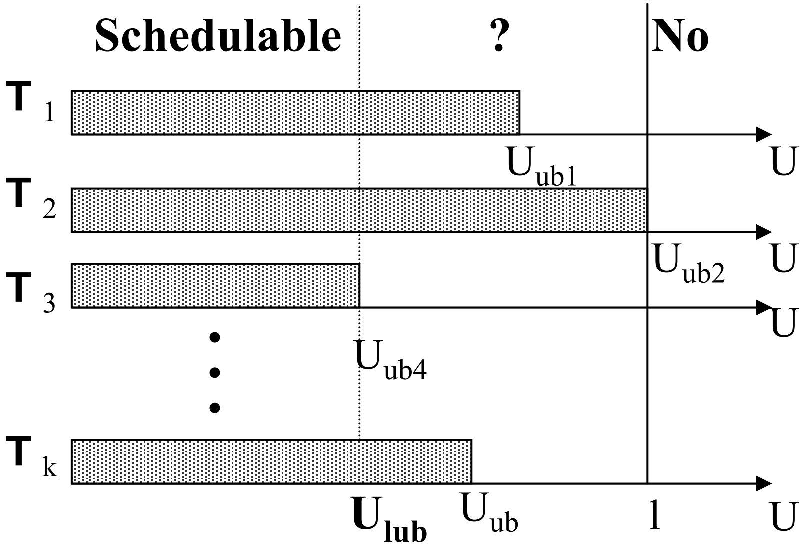
$$U = \sum_{i=1}^{n} \frac{p_i}{\tau_i}$$

Horní hranice $\boldsymbol{U}$ pro rozvržení množiny úloh $\mathcal{T}$ pomocí algoritmu $\boldsymbol{A}$:

$$U_{ub}(\mathcal{T},A)$$

© Z. Hanzálek 2005

Pro daný rozvrhovací algoritmus ***A*** se hodnota $\boldsymbol{U_{ub}}$ mění v závislosti na množině úloh $\mathcal{T}$



Pro daný algoritmus *A* nás zajímá **least upper bound**:

$$U_{lub}(A)=min_{\Gamma}\, U_{ub}(\mathcal{T},A)$$

© Z. Hanzálek 2005

Pro RMS platí „Utilization bound theorem“

Postačující podmínka:

Libovolná množina *n* nezávislých periodických úloh je rozvrhnutelná pomocí RMS jestliže:

$$U(n) \leq n(2^{1/n}-1)$$

n	1	2	3	4	5	6	7	8	9	10
U_{lub}	1,00	0,828	0,780	0,757	0,743	0,735	0,729	0,724	0,721	0,718

$$U_{lub}(RMS) = \lim_{n->\infty} n(2^{1/n}-1) = \ln 2 = 0.69$$

© Z. Hanzálek 2005

Utilization bound theorem - **pesimistický odhad**

Pro příklad 1:

Úloha T_1: $p_1=40$; $\tau_1=100$; $\Rightarrow$ 0.4
Úloha T_2: $p_2=40$; $\tau_2=150$; $\Rightarrow$ 0.267
Úloha T_3: $p_3=100$; $\tau_3=350$; $\Rightarrow$ 0.286

Zatížení procesoru U = 0.4 + 0.267 + 0.28 = 0.953 **překračuje**
Utilization bound theorem jelikož

$$0.953 \not\leq 3(2^{1/3}-1) = 0.780$$

Jelikož T_3 má nižší prioritu než T_1 a T_2 (nemůže ovlivnit jejich vykonávání a je prováděn pouze ve zbylém čase), zkusíme pro 2 úlohy:

T_1 a T_2 **nepřekračuje** $0.667 \leq 2(2^{1/2}-1)=0.828$

Pro úlohu T_3 použijeme „Completion time theorem“

© Z. Hanzálek 2005

Completion time theorem

• podmínka nutná a postačující pro množinu nezávislých periodických úloh pod RMS

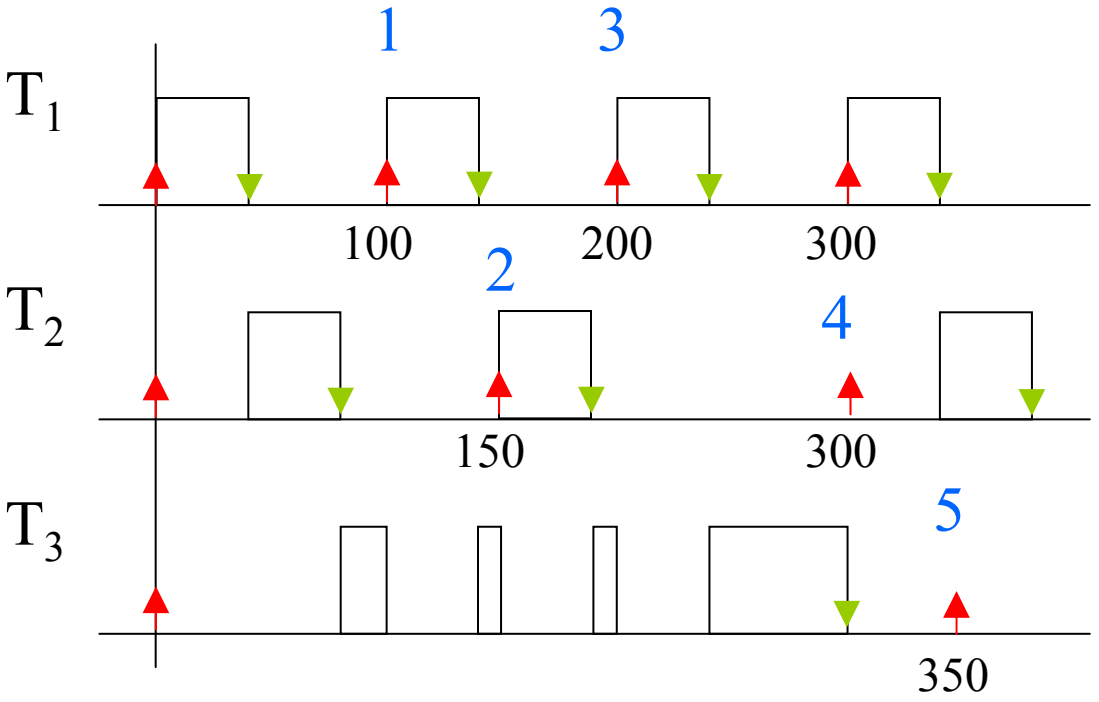
• nejhorší možná situace – **všechny úlohy začínají ve stejný okamžik** (s nulovým posunem fáze). $\Rightarrow$ Stačí vyšetřit pouze **jednu periodu** zkoumané úlohy T_x (v další periodě bude příznivější fáze).

• pro danou úlohu T_x prohlíží všechny okamžiky, odpovídající **konci periody T_x** a **koncům period úloh s vyšší prioritou**. Ostatní okamžiky nejsou z hlediska zpracování úlohy T_x zajímavé.

• v tyto okamžiky kontrolujeme zda byly **dokončeny všechny úlohy tolikrát, kolikrát byly spuštěny**

© Z. Hanzálek 2005

Pro příklad 1:



▲	konec periody (deadline)
▼	dokončení úlohy
1,…, 4	konce period úloh T_1 a T_2
5	dealine úlohy T_3

© Z. Hanzálek 2005

- Pro analýzu **není potřeba odvodit časový průběh** úlohy T_3 (ani úloh T_1 a T_2).
- Úlohu T_3 lze rozvrhnout jestliže platí **alespoň jedna** z následujících nerovnic:

1	$p_1 + p_2 + p_3 \leq \tau_1$	40+40+100>100	NO
2	$2p_1 + p_2 + p_3 \leq \tau_2$	80+40+100>150	NO
3	$2p_1 + 2p_2 + p_3 \leq 2\tau_1$	80+80+100>200	NO
4	$3p_1 + 2p_2 + p_3 \leq 2\tau_2$	120+80+100=300	YES
5	$4p_1 + 3p_2 + p_3 \leq \tau_3$	160+120+100>350	NO

$\Rightarrow$úlohuT_3 lze rozvrhnout jelikož při nejhorším fázování je **dokončena v čase 300**

Pozn: tento prostý součet p_i si můžeme dovolit, jelikož víme že v rozvrhu se od času 0 do tohoto okamžiku neobjeví prodleva (kdyby se objevila, tak by to bylo rozvrhnutelné jelikož procesor spočítal vše co musel i po nejhorším zfázování a přišli bychom na to v některém z předchozích okamžiků)

© Z. Hanzálek 2005

RMS rozšíření pro meziprocesní komunikaci pomocí semaforů

Pokud úlohy sdílí zdroje s exkluzivním přístupem (kritická sekce v kódu), potom úloha, která momentálně drží sdílený zdroj může způsobit **blokování úlohy s vyšší prioritou** čekající na sdílený zdroj.

Pokud je toto blokování omezené, potom můžeme spočítat $\boldsymbol{B_i}$ **nejdelší dobu blokování úlohy**, a vzít ji v úvahu ve zobecněném **utilization bound theoremu** a zobecněném **completion time theoremu**.

…pesimistický odhad

© Z. Hanzálek 2005

Rozšíření RMS pro aperiodické úlohy
- Fixed priority servers

Předpoklad: ani aperiodické úlohy nemohou přicházet libovolně často a nemohou zabírat libovolné množství procesorového času

Řešení:

- na pozadí
- pomocí serverů - s **periodou** τ_S je připraven server (periodická úloha) s přidělenou **kapacitou** p_S

© Z. Hanzálek 2005

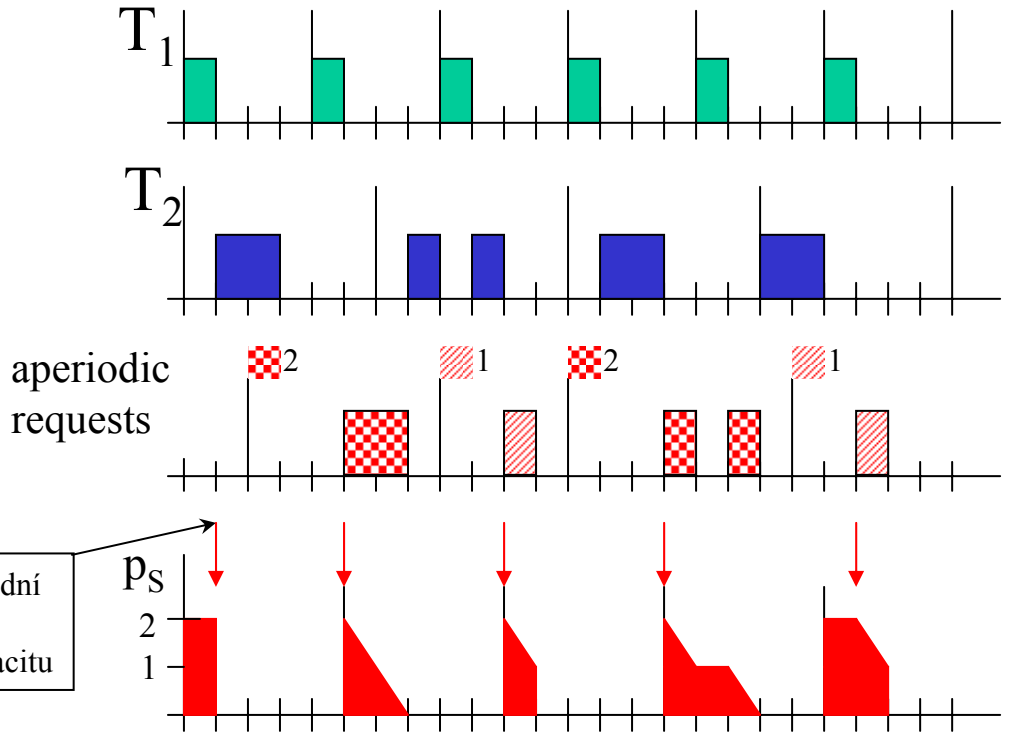
Polling server

- v okamžiku své aktivace obslouží **<u>již připravené</u>** aperiodické úlohy v rámci přidělené **kapacity** $\mathbf{p_S}$. Pokud v tomto okamžiku nejsou žádné připravené, potom se kapacita serveru stornuje (jako číšník který zjistil, že zrovna nikdo nic nepotřebuje).

Příklad 2: dvě periodické úlohy + polling server (priority podle RMS)

	p_i	τ_i
T_1	1	4
T_2	2	6

	p_S	τ_S
server	2	5



zde je spuštěn server (díky tomu, že má střední prioritu), který zjistil,že žádná aperiodická úloha není připravena a stornoval svou kapacitu

© Z. Hanzálek 2005

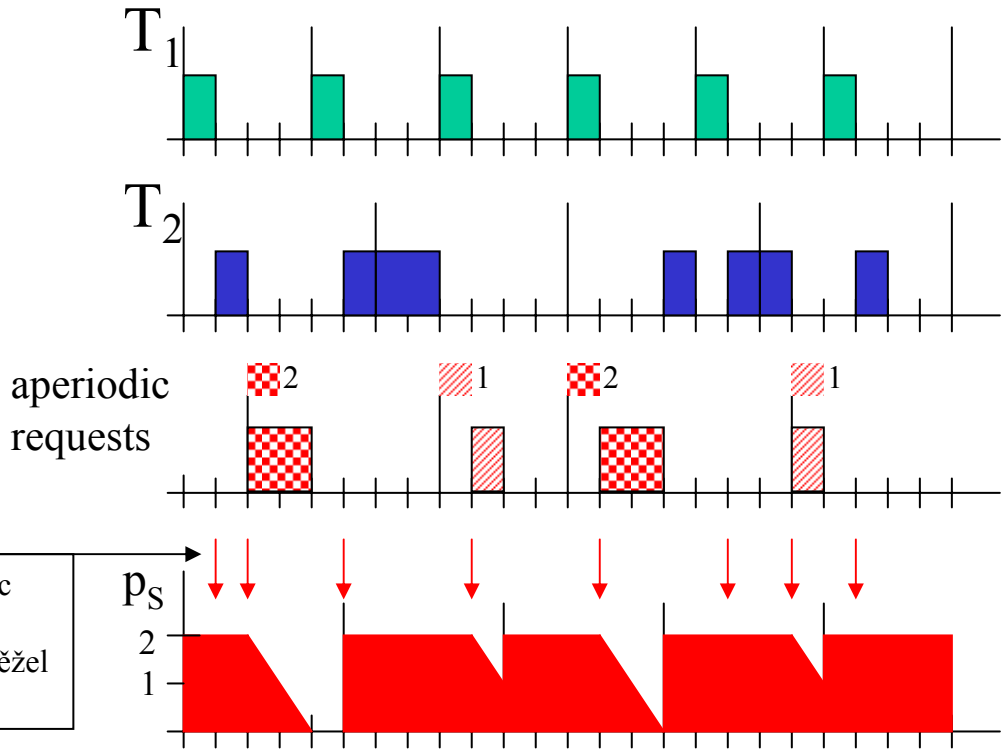
Defferable server

- v rámci přidělené **kapacity p_S** obslouží aperiodické úlohy
- nespotřebovaná kapacita je udržována do konce periody (trpělivý číšník)

Př. 2: dvě periodické úlohy + defferable server (priority podle RMS)

	p_i	τ_i
T_1	1	4
T_2	2	6

	p_S	τ_S
server	2	5



server je spuštěn, pokud má kapacitu a navíc buďto skončil proces s vyšší prioritou nebo přišel ´aperiodic reguest´ ve chvíli, kdy neběžel proces s vyšší prioritou

© Z. Hanzálek 2005

RMS - závěr

• Rate monotonic teorie je užitečný nástroj pro určení priorit úloh pro preemptivní jádro na základě **časových parametrů** aplikace.

• Rozumné **chování při přetížení** (nepředpokládané prodloužení p_i) – deadline je nejprve překročena u úlohy s nejnižší prioritou.

• Rozšíření pro **úlohy se synchronizací** – prodlužuje vykonávání úlohy o nejdelší předpokládané doby čekání na semafory … příliš pesimistické.

© Z. Hanzálek 2005

EDF pro on-line rozvrhování

- EDF může být použito nejen pro off-line (známé parametry úloh včetně release time) rozvrhování ale i pro on-line rozvrhování (momentálně běží ta úloha, jejíž deadline je nejblíže). Také to lze chápat jako dynamické přiřazování priorit.
- **optimalita**: jelikož EDF není vázáno na periodicitu úloh, tak jeho optimalita dokázaná pro aperiodické úlohy, **platí i pro periodické úlohy**
- Theorem: **Množina nezávislých úloh je rozvrhnutelná pomocí EDF právě když:**

$$U = \sum_{i=1}^{n} \frac{p_i}{\tau_i} \leq 1$$

© Z. Hanzálek 2005