Hierarchical Token Bucket

Martin Devera

CDI computers s.r.o.

Email: devik@cdi.cz

http://luxik.cdi.cz/~devik/qos/htb/

Cíle QoS

- Zajistit *kvalitu* služby. Kvalitou rozumíme garantovaný tok a odezvy.
- Pro TCP je odhad datového toku pro danou ztrátovost

$$Bandwidth = 1.22 \frac{ATU}{RTT\sqrt{Loss}} \; pro \; Loss \lesssim 5\%$$

je evidentní, že tok můžeme ovlivnit změnou

  - ztrátovosti v poměrně malém rozsahu
  - RTT zpožděním paketů
  - ATU (aktuální velikostí bloku)

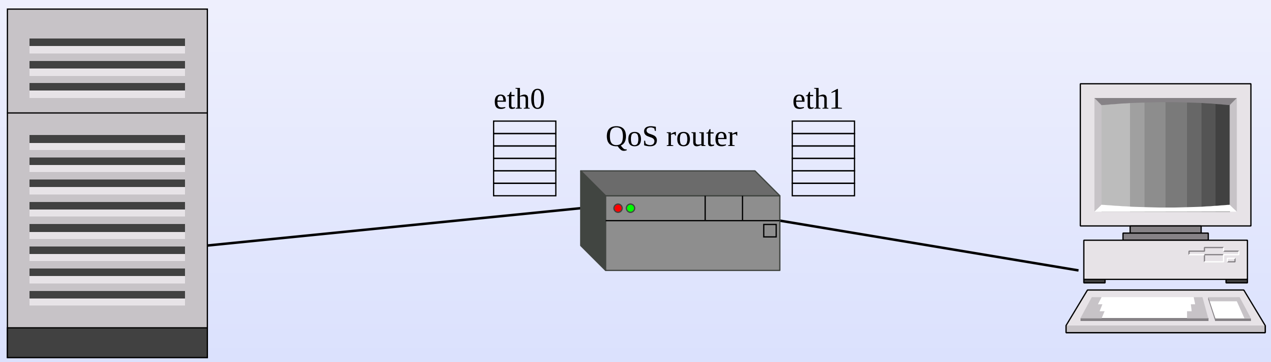
Je nutno uvažovat snahu TCP stacku kompenzovat změny RTT změnami ATU.

Prostředky realizace QoS

- Modifikace paketu (packeteer) - proti současným standartům, v budoucnu možná využití ECN.
- Zahazování paketu (bez frontování) - použitelné pro ingress, komplikovaná detekce backlogovaného toku.
- Frontování (se zahazováním) - nejlepší současná cesta, problémy s ingresem.

QoS a frontování

- Router mívá výstupní frontu paketů, kde může kontrolovat pořadí odchozích paketů.
- Není jednoduché ovlivňovat příchozí tok paketů - je možno je pouze zahodit.



Fronta jako nástroj QoS

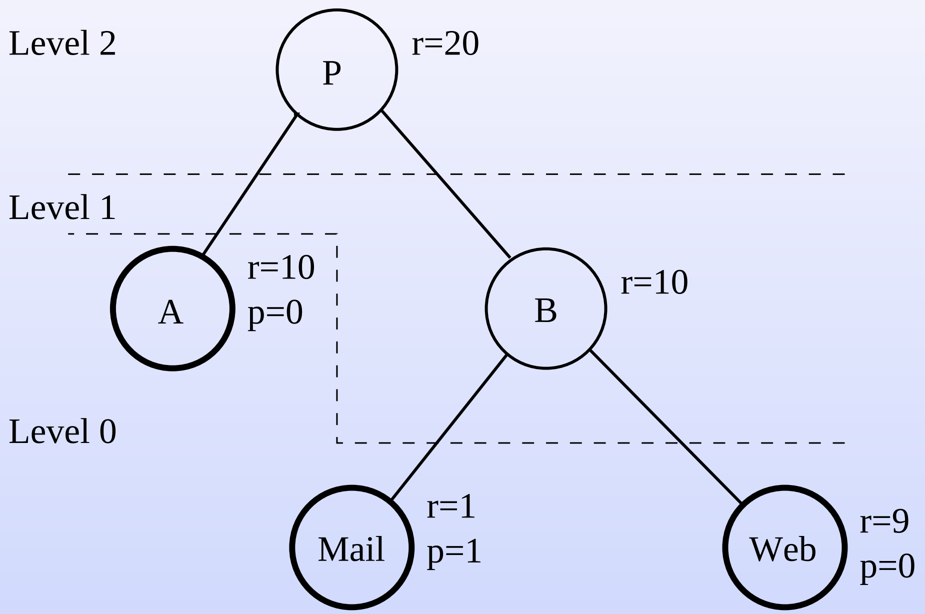
- Zahození paketu při jeho příchodu do plné fronty (FIFO, SFQ).
- Zahození na základě expirace paketu.
- Zdržení paketu (TBF).
- Změna pořadí paketů (PRIO).
- Zdržení a změna pořadí paketů (CBQ, HTB).

Třídy

- Pro pohodlnější práci řadíme jednotlivé pakety do *tříd* podle adres, typů služeb, velikosti apod.
- Řazení provádí *klasifikátor* a to
  - při vstupu paketu do interface a jemu přiřazené QoS disciplíny na každém routeru.
  - při vstupu paketu do sítě pod společnou správou (tzv. *Diffserv*).
- QoS disciplíny berou třídu jako celek skládající se z již atomických paketů.

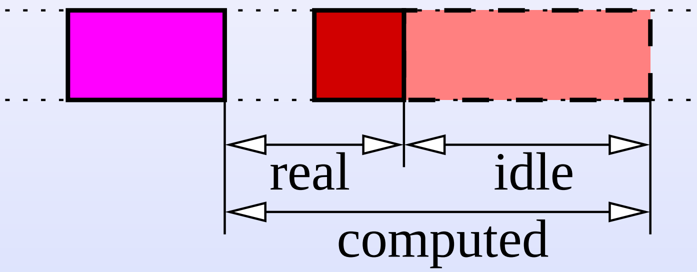
Hierarchie tříd

- Třídy je často sdružují do nadtříd, a tak vzniká hierarchie tříd.
- Každá třída má přiřazený garantovaný tok r a listy i prioritu p (vyšší priorita má menší číselnou hodnotu).
- Priority odráží očekávané zpoždění paketů.



Měření toku - estimátor CBQ

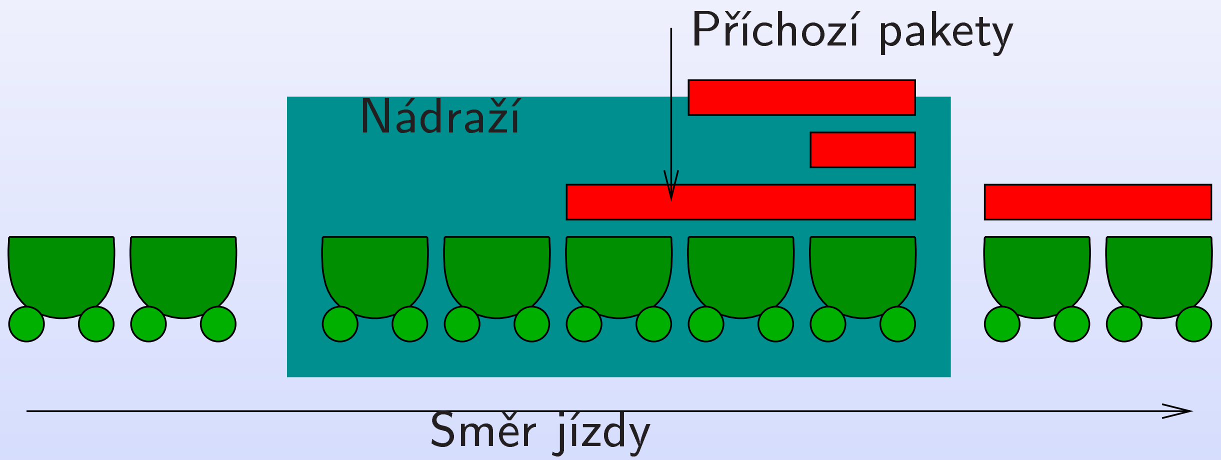
- Je třeba znát, zda je třída nad či pod svým limitem.
- CBQ dokument a všechny implementace pracují s průměrováním tzv. *idle* času a jeho porovnávání s nulou.
- Problémy s přesností určení $real$ času a závislost na fyzické rychlosti rozhraní (média).



$$computed = \frac{red\_packet\_size}{class\_rate}$$

Leaky Bucket estimátor

- Určíme $rate$ (rychlost příjezdu jednobajtových vozíků) a $burst$ (velikost nádraží).
- Minimální velikost $burst \geq rate \cdot \Delta$, kde Δ je rozlišení systémových hodin - jinak by se nedalo stihnout odbavit vozíky.



Leaky Bucket vs. Idle estimátor

Leaky Bucket

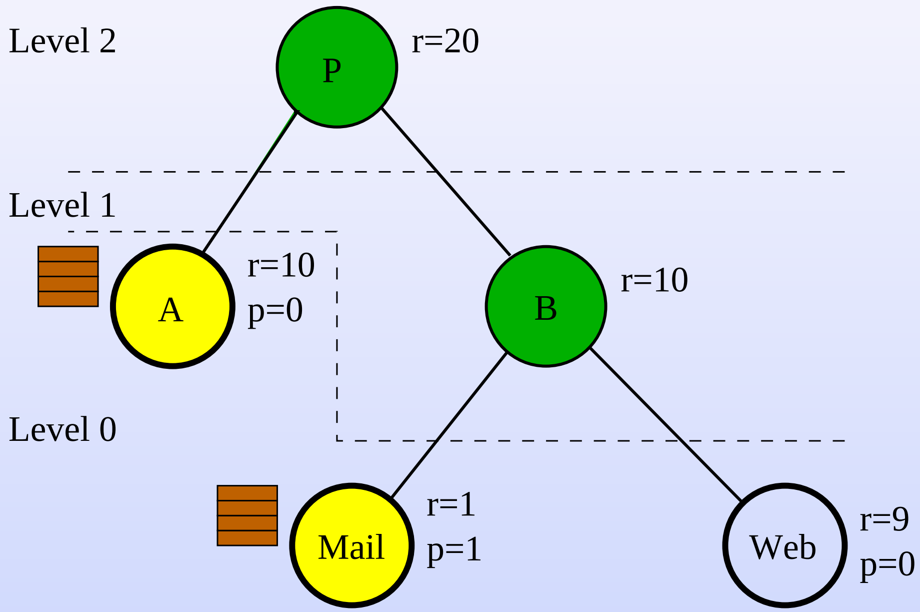
- se jednoduše a efektivně implementuje,
- je intuitivní pro nastavení,
- nezávislý na fyzické rychlosti rozhraní,
- nevyžaduje přesný systémový časovač.

Idle estimátor

- produkuje hladší tok s možností nastavení plynulosti.

CBQ

- Podle priorit by měla další paket poslat třída A, ale pak by třída B nedostala garantovaný tok.
- CBQ zavádí *toplevel*, což je level nejvyššího nespokojeného uzlu (zde level 1) a ze tříd výše se nesmí půjčovat.

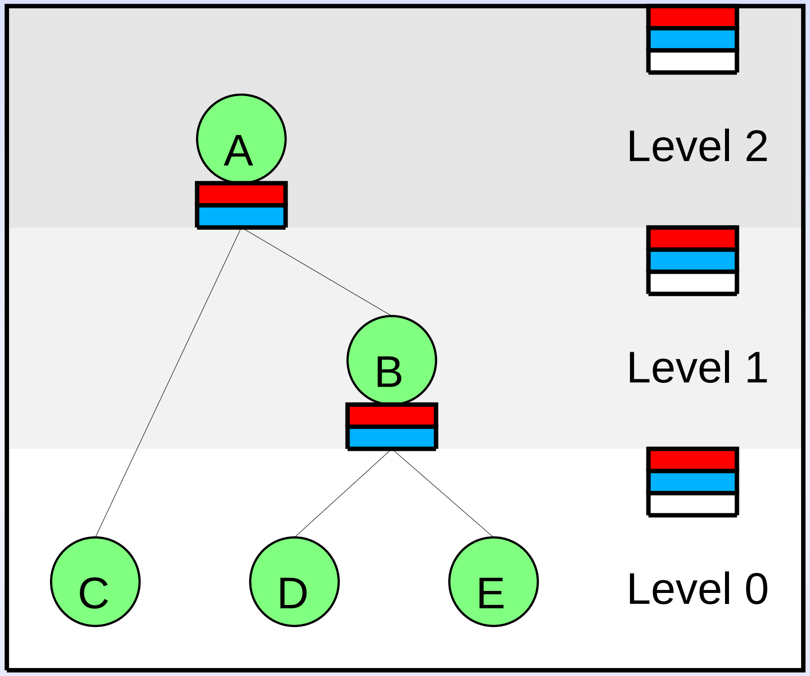


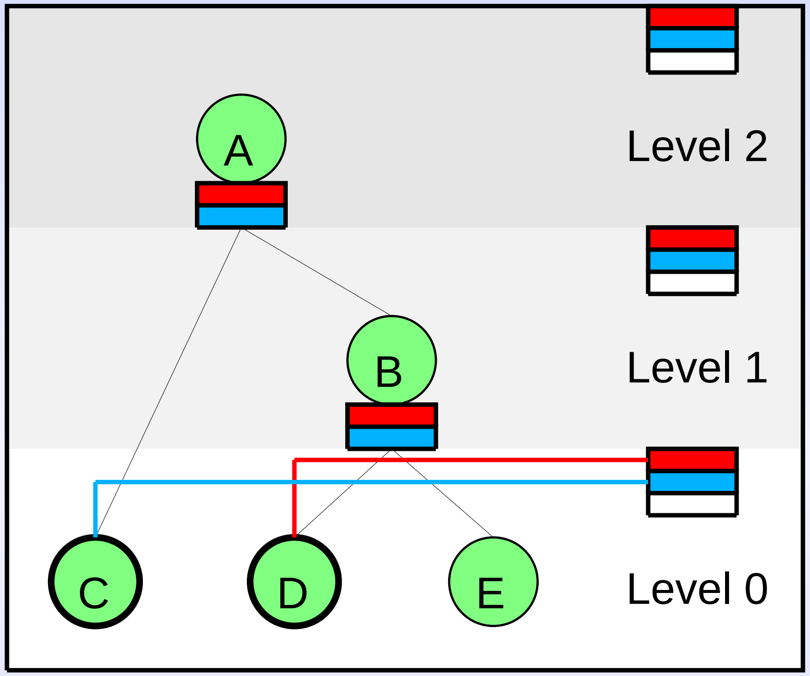
Problémy sdílení CBQ

- Definovali jsme *formální pravidla* CBQ sdílení. Cíle CBQ budou naplněny, pokud bude vždy známa přesná hodnota toplevelu.
- Implementace používají *aproximaci* - globální proměnná toplevel je upravována při zafrontování a odebrání paketu $\Longrightarrow$ chyby ve sdílení (nesnížení toplevelu při *asynchronním* přechodu pod limit).

Implementace toplevel pravidla v HTB

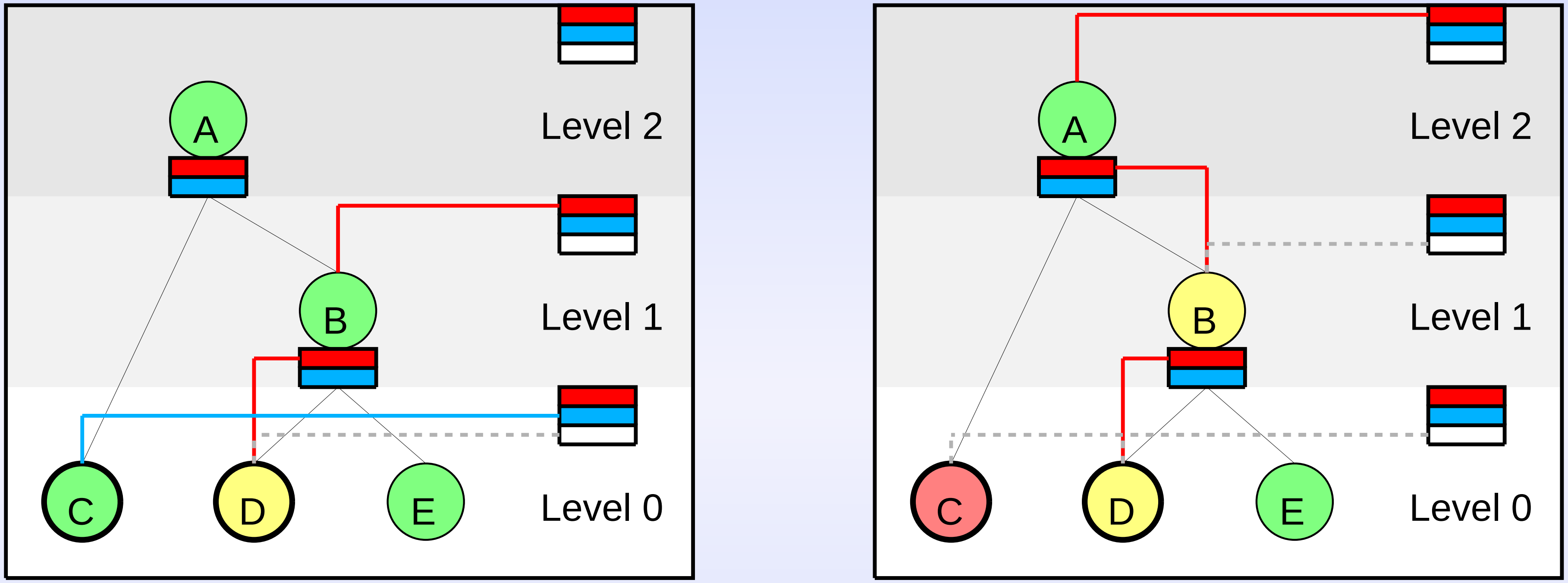
- Nový parametr toku - *ceil* neboli maximální povolený tok.
- Definujeme barvy tříd, červená pro aktuální tok $a > ceil$, žlutá pro $ceil \geq a > rate$ a zelená pro $a \leq rate$.
- Informace o toplevelu je zakódována prvním neprázdným globálním půjčovacím seznamem.
- Globální seznam udržuje třídy jednoho levelu a priority, které mohou odeslat paket.
- Cesta k listu, který paket vlastní, je tvořena lokálními půjčovacími seznamy.
- Pro balancing jsou seznamy sortovány pomocí RB-tree s classid jako klíčem.





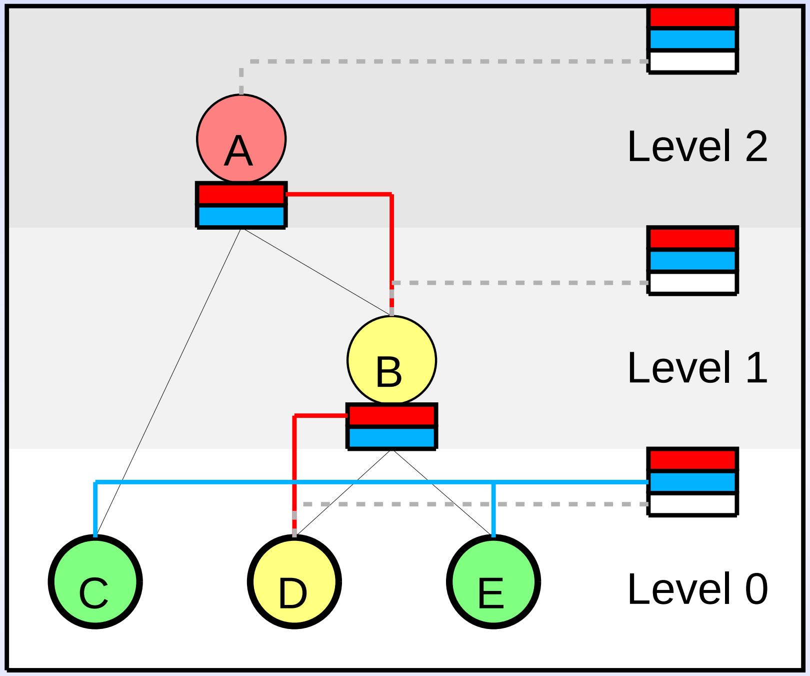
1. Žádné pakety, třídy jsou *zelené*, pouze D má vyšší prioritu.

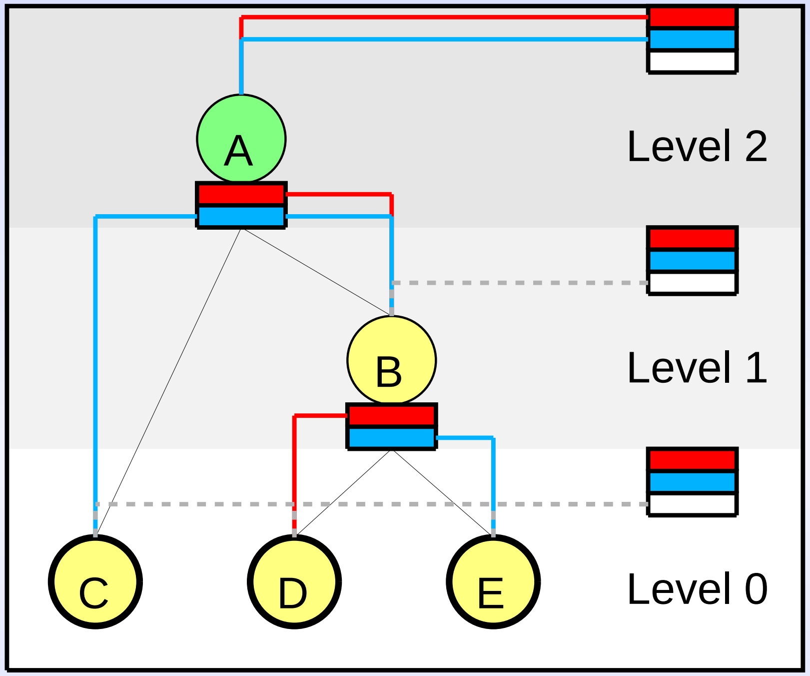
2. Přišly pakety pro C a D.



1. D je nyní *žlutá*.

2. C je *červená* a zároveň B je teď *žlutá*.





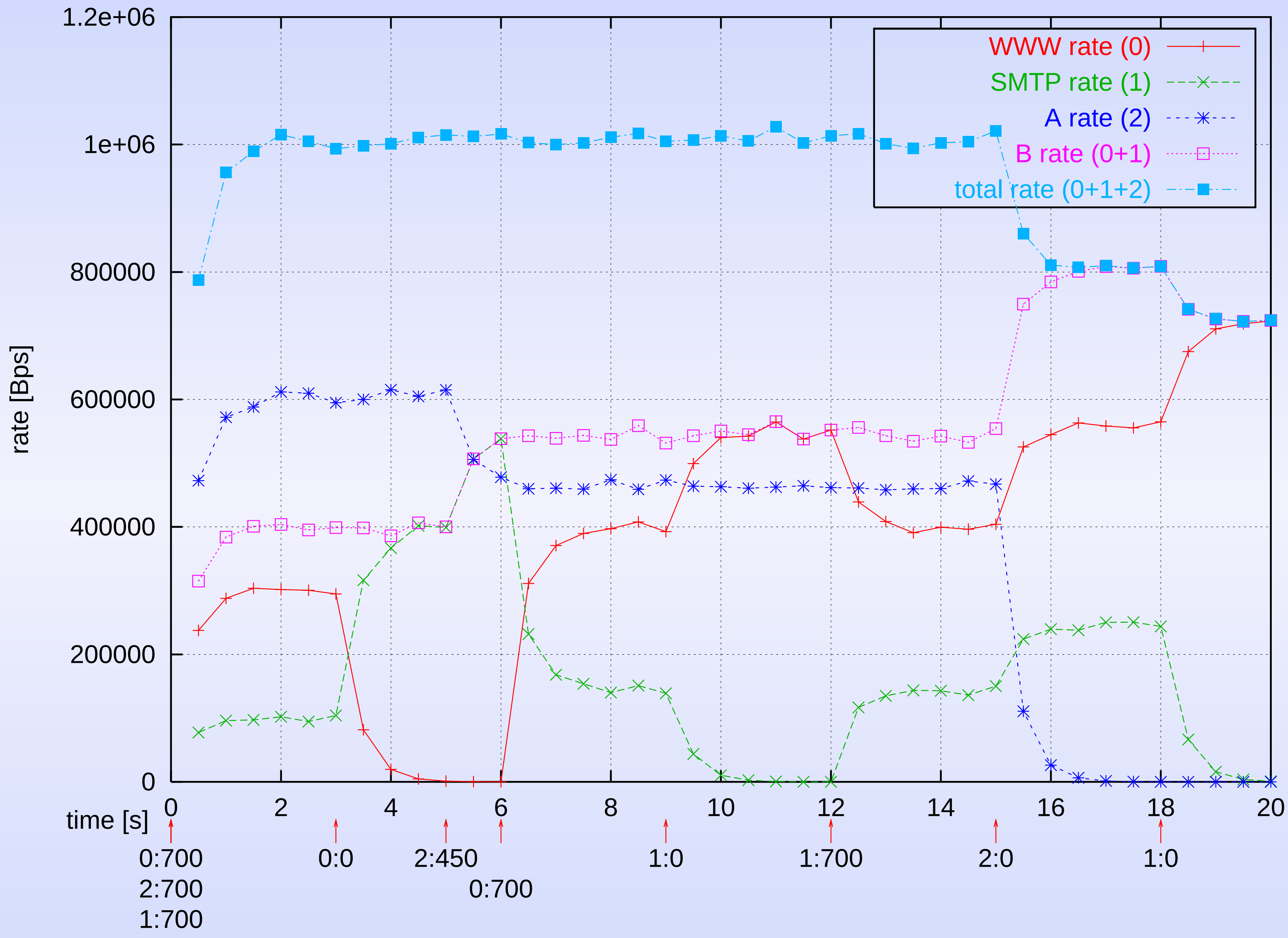
1. A je *červená*, a C i E jsou *plné* a *zelené*.
2. Všechny listy si půjčují od A.

Shrnutí algoritmu HTB

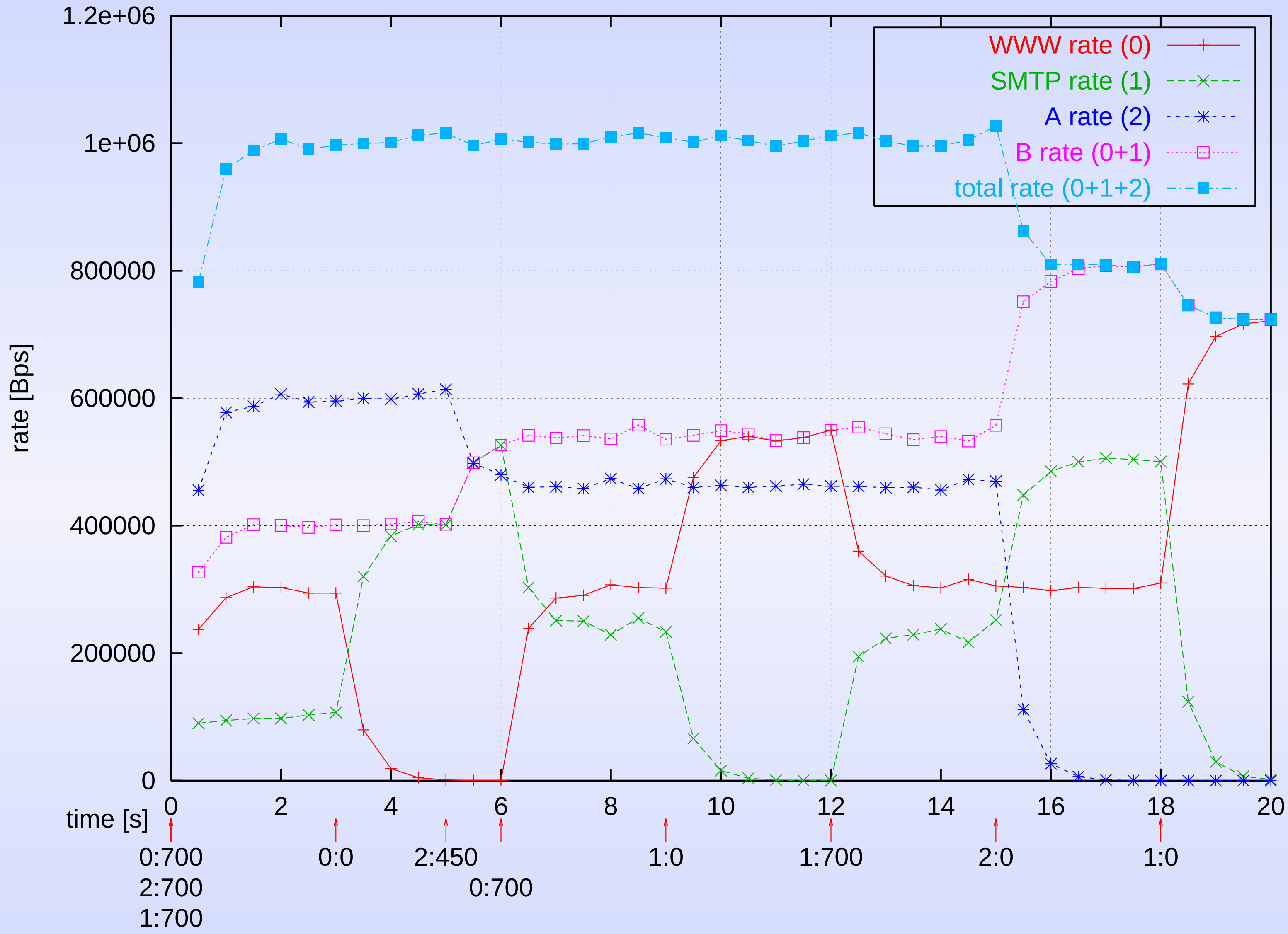
- Složitost je logaritmická per paket.
- Žádná aproximace - scheduling je naprosto exaktní.
- Možnost nastavení stropu.
- **HTB je tedy typem CBQ !**

Příklad nastavení

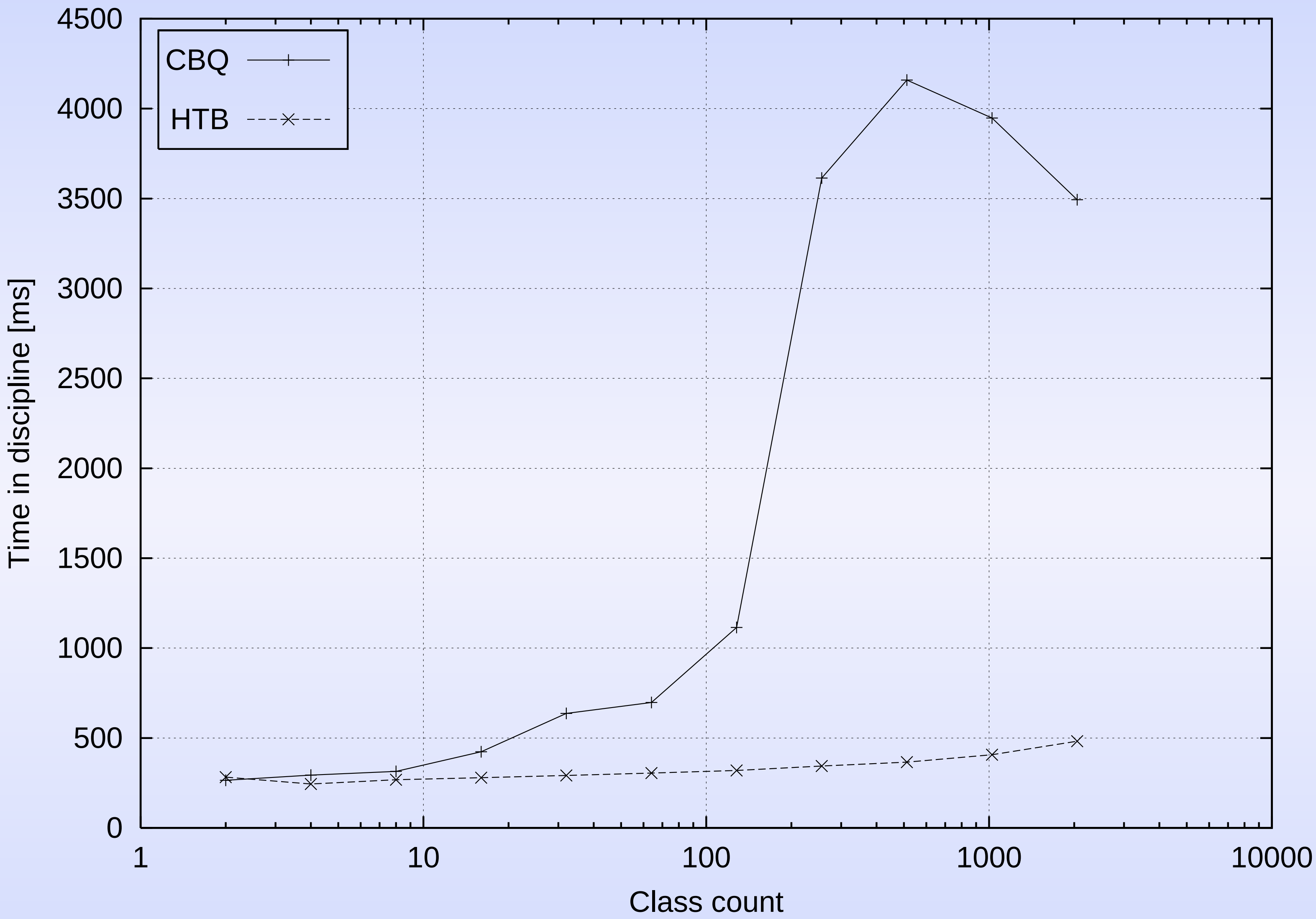
```
tc qdisc add dev eth0 root handle 1: htb default 12
TCC="tc class add dev eth0"
TCC parent 1: classid 1:1 htb rate 1000kbps ceil 1000kbps
TCC parent 1: classid 1:2 htb rate 400kbps ceil 800kbps
TCC parent 1:2 classid 1:10 htb rate 300kbps ceil 1000kbps prio 1
TCC parent 1:2 classid 1:11 htb rate 100kbps ceil 1000kbps prio 1
TCC parent 1:1 classid 1:12 htb rate 600kbps ceil 1000kbps prio 1

tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 \
   match tcp sport 80 0xffff flowid 1:10
tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 \
   match ip dst 1.2.3.4 match tcp dport 25 flowid 1:11
```



Toky pro rate A=600k, B=400k, SMTP=100k, WWW=300k.



Toky s vyšší prioritou u SMTP (Mail) třídy.



- CBQ rychle naráží na limit 5ti sekundového běhu.