



dsn

X36PKO
QoS
Jiří Smítka

QoS - co, prosím ?

- Quality of Services = kvalita služeb
- Opatření snažící se zaručit koncovému uživateli doručení dat v potřebné kvalitě
- Uplatňuje se v přenosu multimédií, IP telefonii, atd.
- Kvalita služby je ovlivněna:
 - stanicemi (uživatelé, servery)
 - směrovači, prepínači
 - linkami (mezi směrovači, LAN)

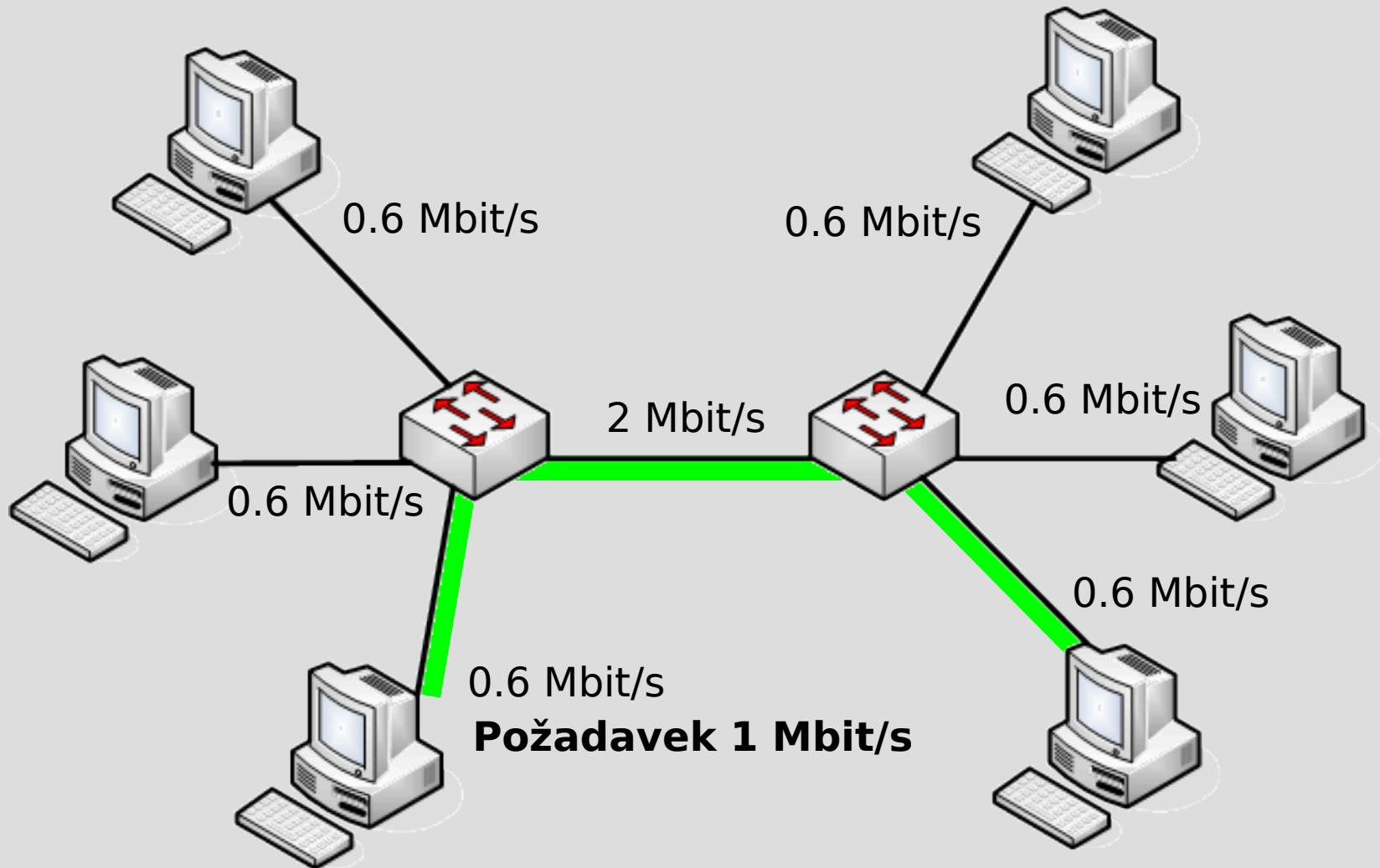
Sdílení kapacity sítě

- V jednoduché síti typu Internet se všichni uživatelé dělí o prostředky sítě stejným dílem
- 100 uživatelů + linka 100 Mbit/s
=> 1 Mbit/s na jednoho uživatele
- Většinou není menší rychlost problém
- Některé aplikace (např. IP telefonie) však nemusí fungovat

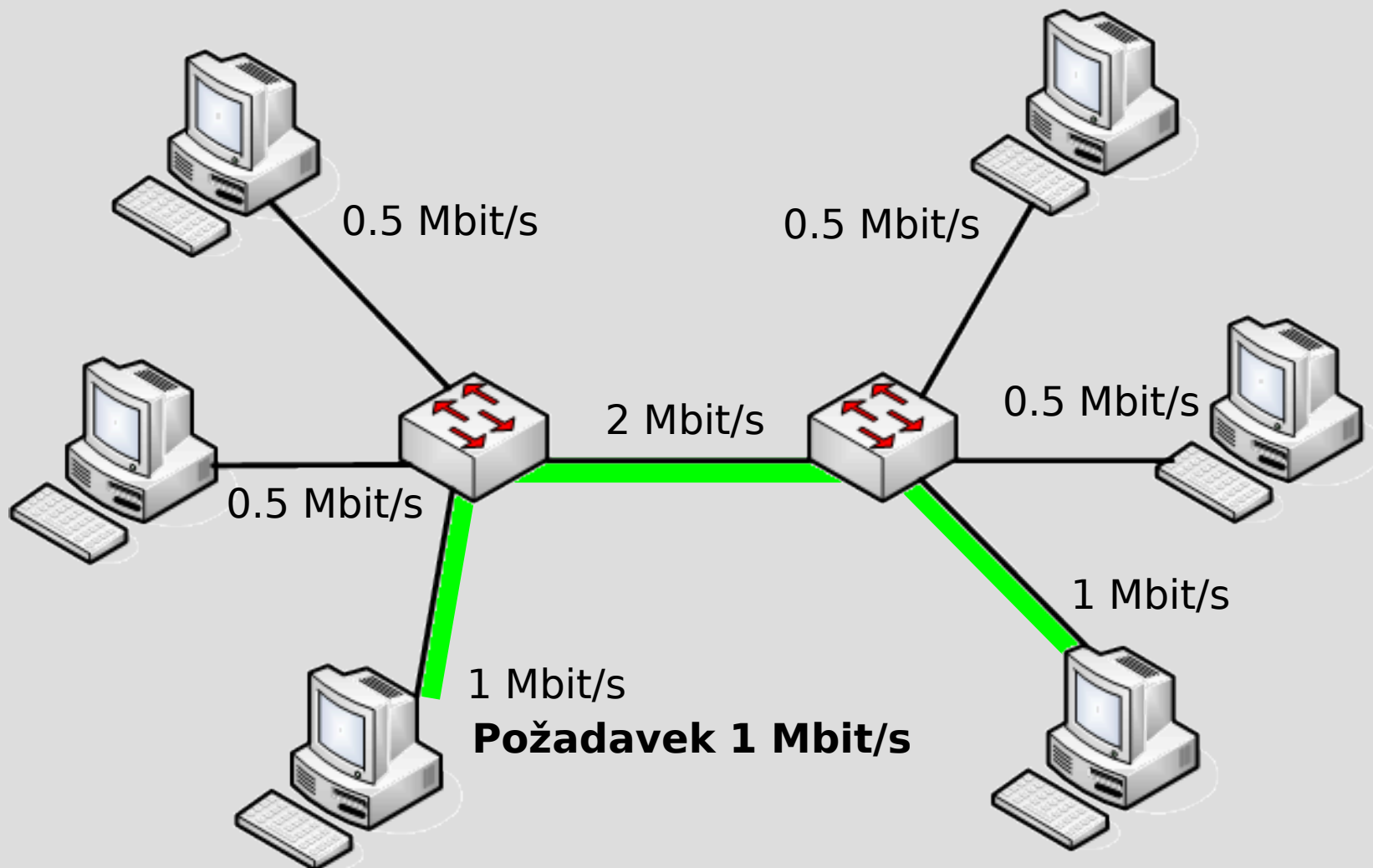
Možnosti QoS

- Rezervovat přenosovou kapacitu pro daný kanál
- Nastavit vyšší prioritu některým službám (např. ssh) a zkrátit jejich odezvu
- Omezit přenos na definovaný limit (např. omezení FTP, aby bylo možno přistupovat na WWW)
- Definovat maximální zpoždění dat

Příklad sítě bez QoS



Příklad sítě s QoS



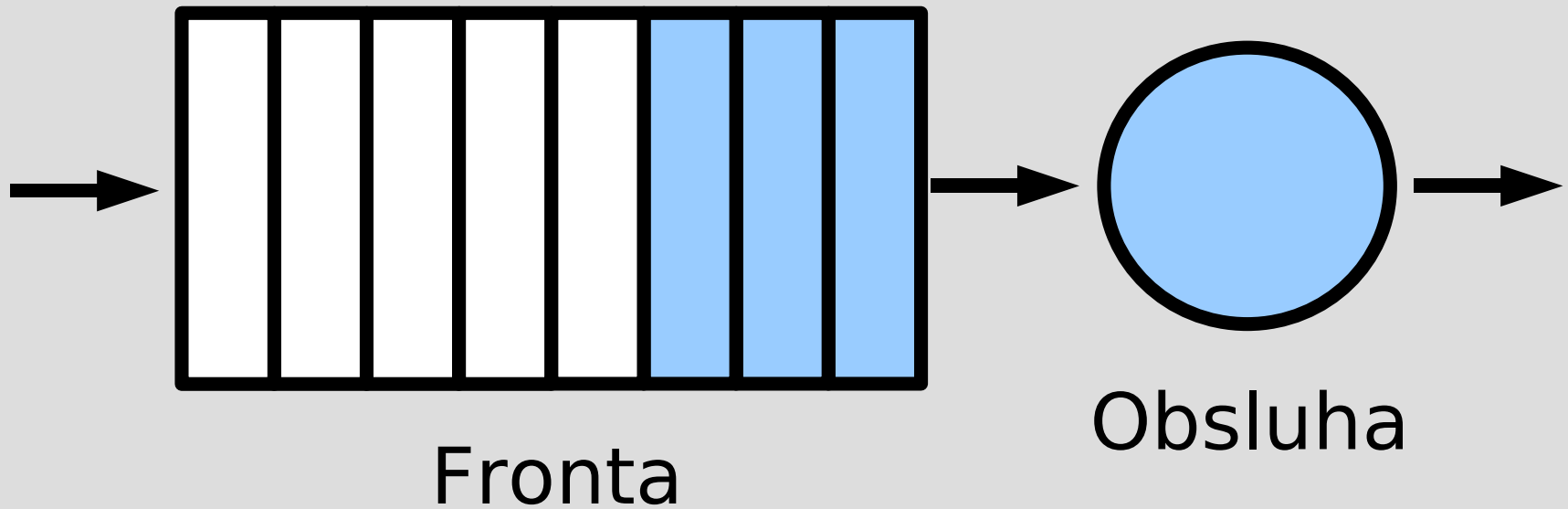
Parametry tvořící QoS

- šířka pásma = rychlost přenosu dat
- jednosměrné zpoždění
 - čas potřebný pro přenesení paketu přes fyzické médium
 - čas způsobený řazením do front
- rozptyl zpoždění
- ztrátovost paketů

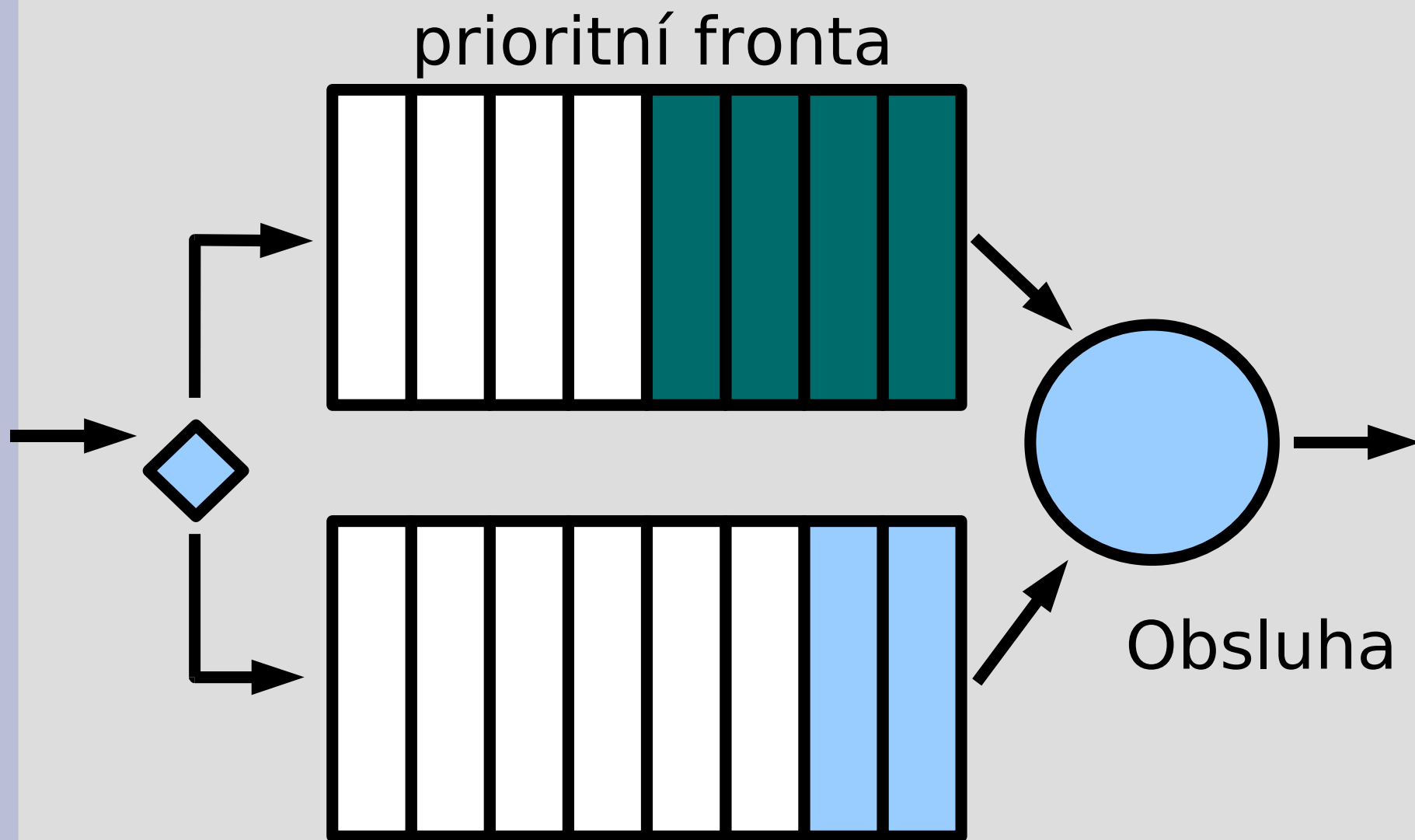
Plánovací mechanismy

- FIFO
- Prioritní FIFO
- Round Robin
- WFQ
- Leaky Bucket
- Leaky Bucket + WFQ

FIFO



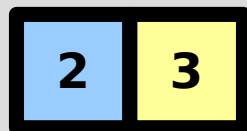
Prioritní FIFO



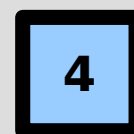
Round Robin

2 fronty (prioritní + ostatní)

Příchod

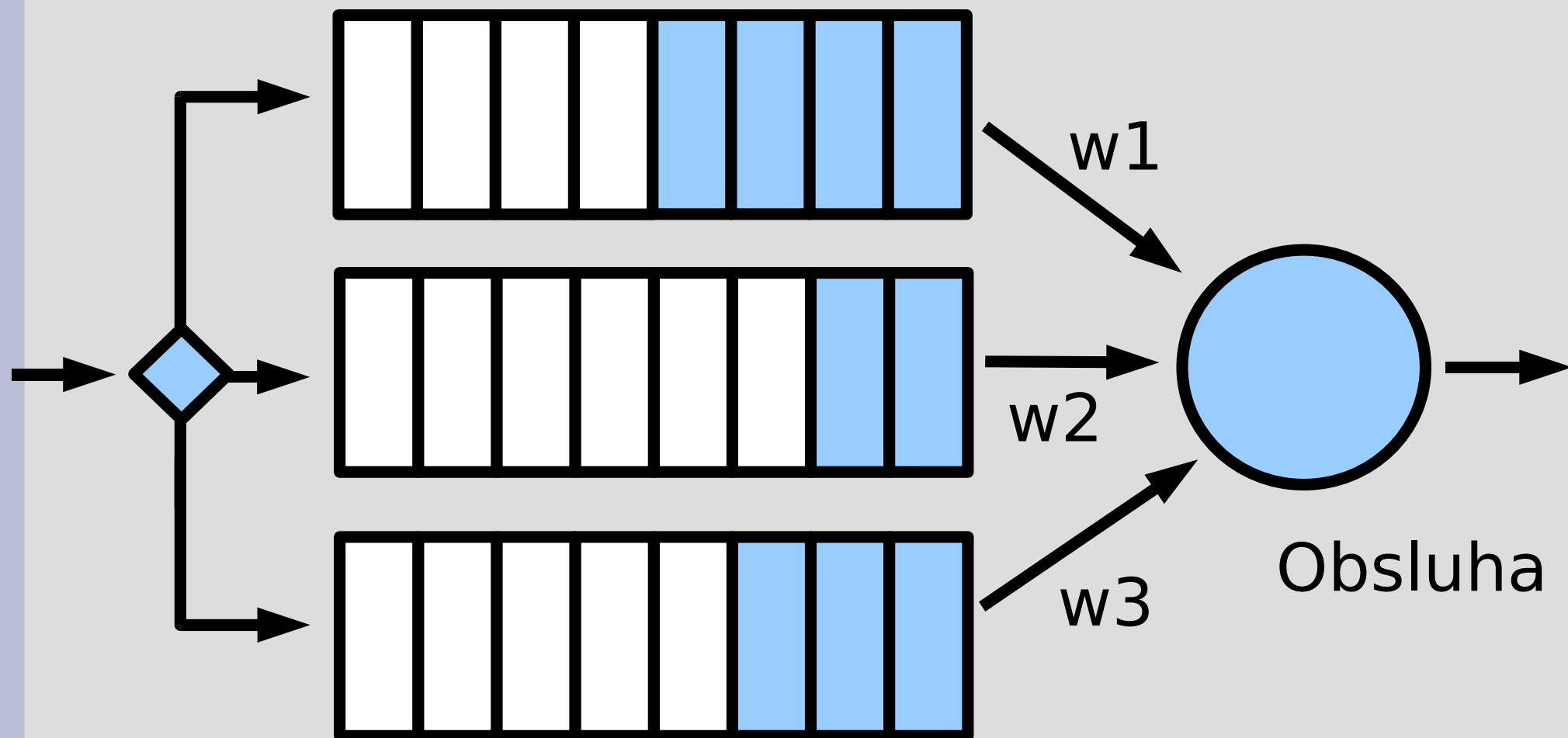


Obsluha

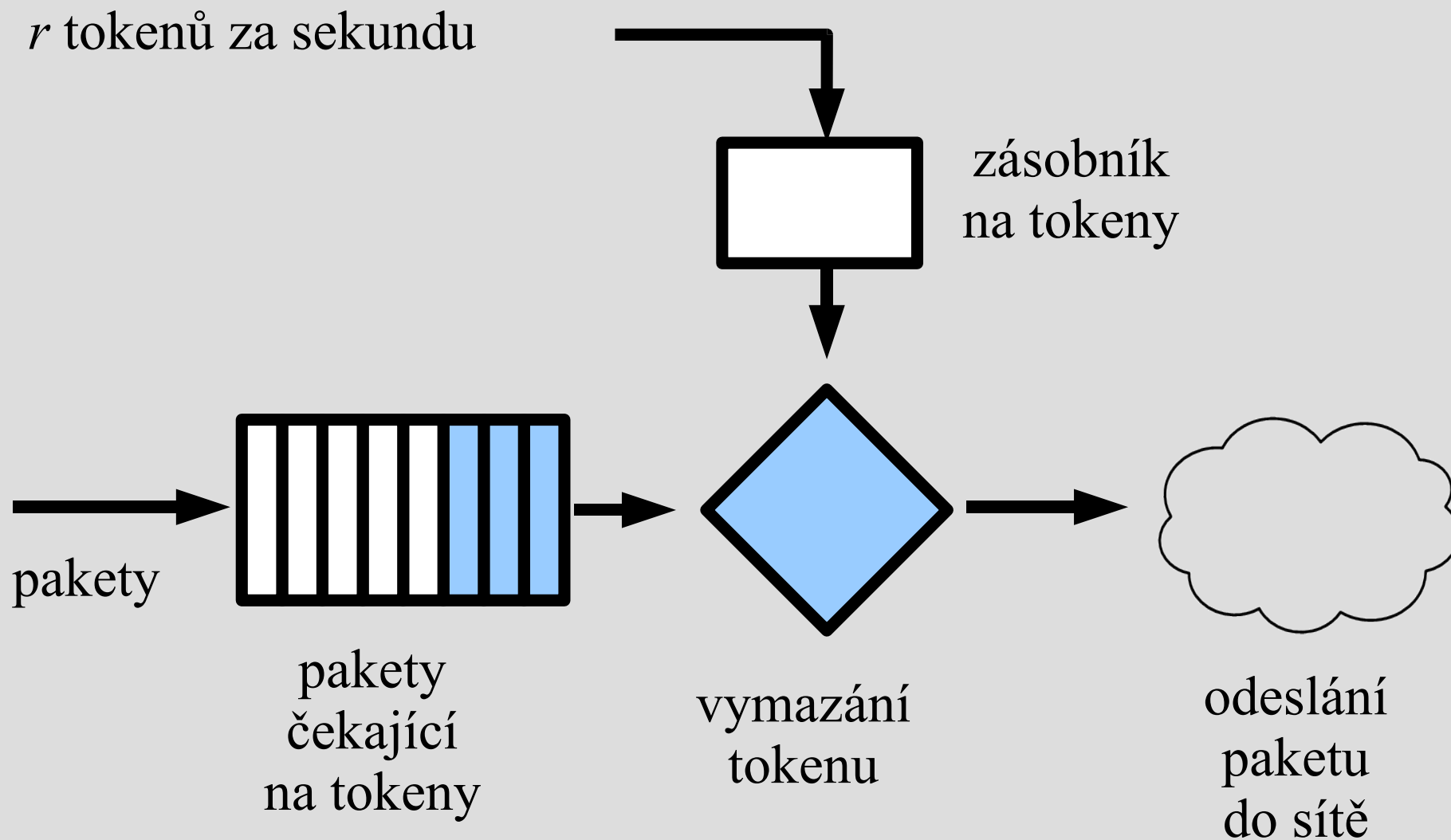


čas

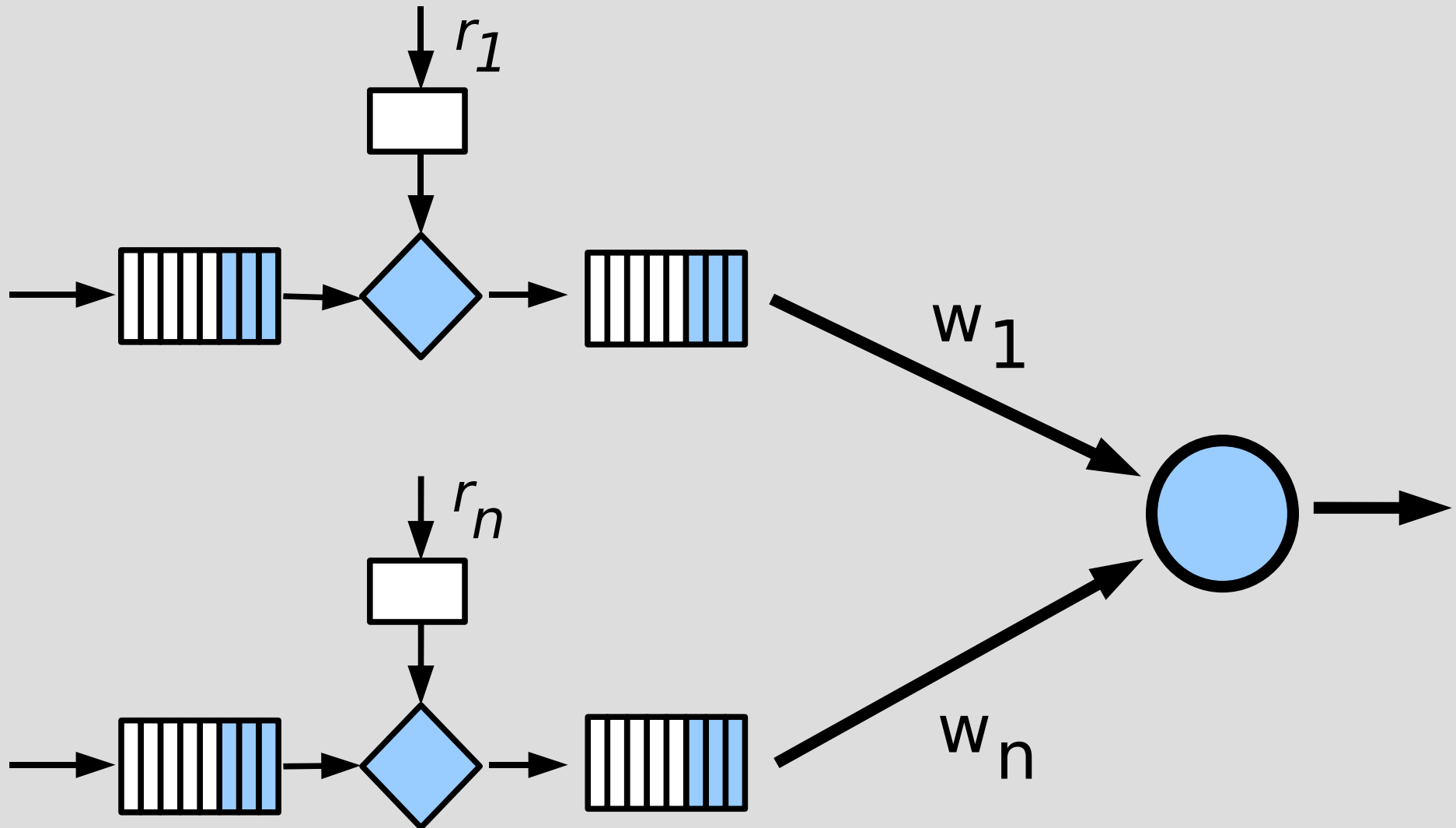
WFQ (weighted fair queuing)



Leaky Bucket



Leaky Bucket + WFQ



Co na to IP protokol ?

- IPv4 má v hlavičce informační pole TOS (Type of Service):

1000	minimalizuj zpoždění
0100	maximalizuj propustnost
0010	maximalizuj spolehlivost
0001	minimalizuj finanční náklady
0000	normální služba

ToS v aplikacích

- Telnet = 1000 (*zpoždění*)
- FTP control = 1000 (*zpoždění*)
- FTP data = 0100 (*propustnost*)
- DNS
 - UDP query = 1000(*zpoždění*)
 - TCP query = 0000
 - Zone transfer = 0100 (*propustnost*)
- ICMP
 - errors = 0000
 - request = 0000
 - response = 0000

TCP a řízení toku

- Efektivně lze omezovat pouze odchozí tok dat, pro příchozí tok musíme použít nepřímé metody.
- Můžeme pozdržet potvrzení a tím prodloužit RTT (round trip time) (pozor na timeout, který způsobí opětovné odeslání dat).
- Můžeme měnit velikost okénka.

RTT v TCP (1)

- RTT (round trip time) je doba, za kterou přijde potvrzení odeslaného datového segmentu
- Od RTT se odvozuje timeout
- Jenže jak dopředu vědět hodnotu RTT ?

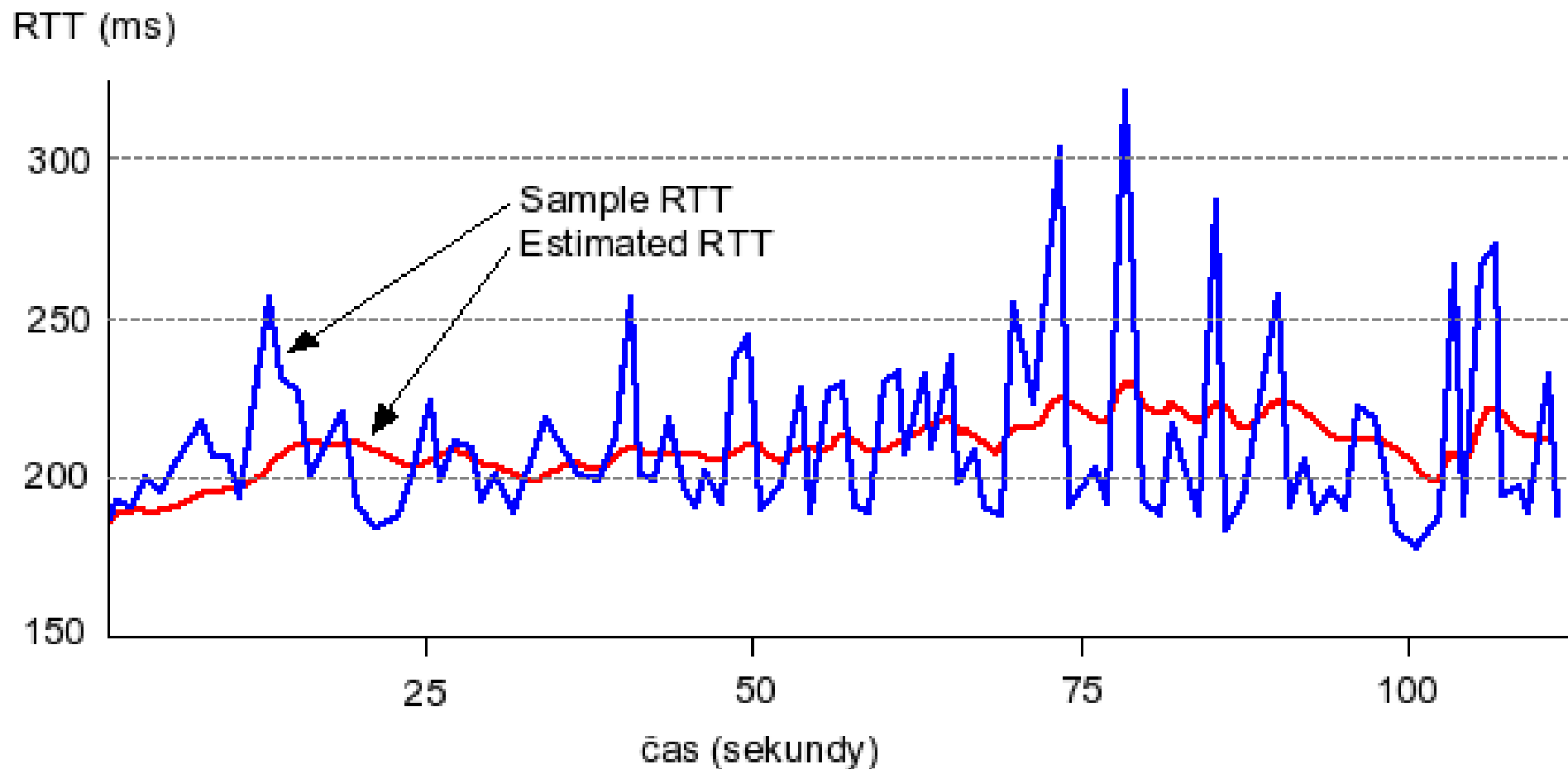
SampleRTT = poslední naměřený RTT

EstimatedRTT = očekávaný RTT

$$\text{EstimatedRTT} = (1-\alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT}$$

$$\alpha = 0,125$$

RTT v TCP (2)



TCP - Timeout

$$\text{DevRTT} = (1-\beta) \cdot \text{DevRTT} + \beta \cdot | \text{SampleRTT} - \text{EstimatedRTT} |$$

(popisuje proměnlivost RTT, $\beta=0,25$)

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 \cdot \text{DevRTT}$$

Timeout se po každé ztrátě paketu zdvojnásobí.

TCP okénko I.

- „slow start“: CongWin = 1 MSS
(MSS = maximum segment size)
- po každém RTT se CongWin zdvojnásobí (s každým paketem roste o 1 MSS)
- V proměnné *Threshold* je uložena hodnota, kde „slow start“ končí a začíná fáze „congestion avoidance“ (zamezení zahlcení)

TCP okénko II.

- Ve fázi „congestion avoidance“ roste CongWin lineárně vždy o 1 MSS.
- Při příjmu 3 duplicitních ACK (detekce ztráty) se Threshold a CongWin nastaví na polovinu poslední hodnoty CongWin.
- Timeout způsobí nastavení Threshold na polovinu CongWin a nastaví CongWin na 1 MSS. Poté zahájí „slow start“.

IPv6 a QoS

- QoS v dnešním Internetu funguje nepovinně, jen v jeho částech.
- Zajištění QoS mezi koncovými uživateli je tedy obtížné.
- IP protokol verze 6 by měl tento nedostatek odstranit
- IPv6 má v hlavičce nová pole definující způsob zpracování a identifikace informací přenášených v síti