Výlet za hranice TCP: protokoly pro sítě s velkým součinem latence a šířky pásma

Petr Holub
hopet@ics.muni.cz

SitSem
2011–10–12

Přehled přednášky

- Tradiční TCP a jeho problémy
- Vylepšení TCP
  - Víceproudové TCP
  - Web100
- Konzervativní rozšíření TCP
  - GridDT
  - Scalable TCP, High-Speed TCP, H-TCP, BIC-TCP
- Rozšíření TCP s podporou IP
  - QuickStart, E-TCP, FAST
- Implementace v OS
- Přístupy odlišné od TCP
  - tsunami
  - RBUDP
  - XCP
  - SCTP, DCCP, STP, Reliable UDP, XTP
- Závěrečné poznámky

Protokoly pro zajištěný přenos dat

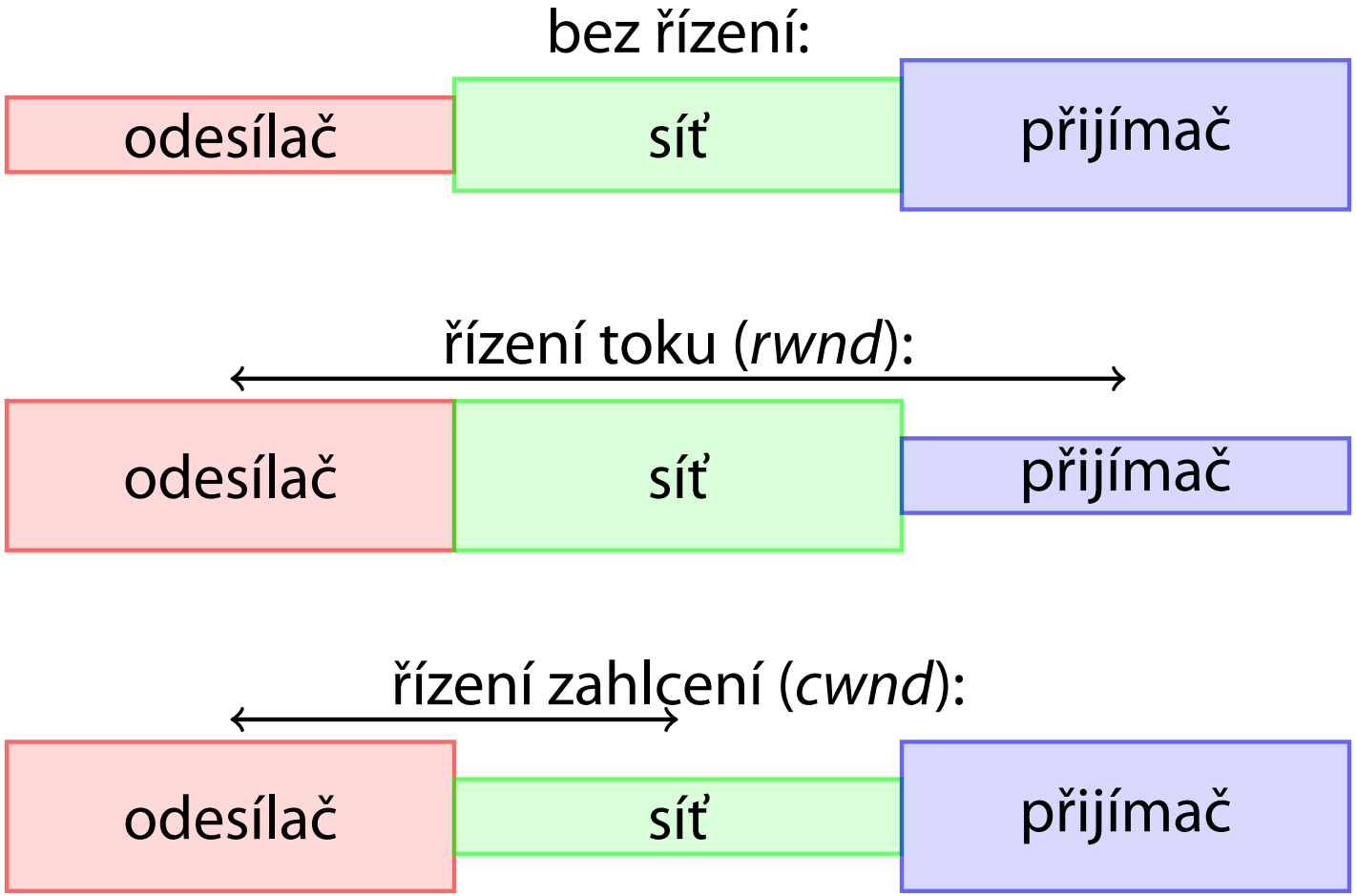
- Zajištění spolehlivosti přenosu
  - retransmise ztracených dat
  - vypomoci může FEC
- Ochrana před zahlcením
  - vysílače, sítě, přijímače
- Hodnocení chování
  - *agresivita* – využití dostupné kapacity
  - *responsivita* – schopnost zotavení se po výpadku
  - *férovost* – získání férového podílu na síťové kapacitě při více účastnících

Problém

- Síťové spoje s *vysokou kapacitou* a *vysokou latencí*
  - iGrid 2005: San Diego $\leftrightarrow$ Brno, RTT $=$ 205 ms
  - SC|05: Seattle $\leftrightarrow$ Brno, RTT $=$ 174 ms
- Tradiční TCP není připraveno pro takové prostředí
  - 10 Gb/s, RTT $=$ 100 ms, 1500B MTU
    $\Longrightarrow$ vysílací okno 83.333 paketů
    $\Longrightarrow$ ztráta jednoho paketu za 1:36 hodiny
- *Jak dosáhnout lepšího využití sítě?*
- *Jak zajistit rozumnou koexistenci s tradičním TCP?*
- *Jak zajistit postupné nasazování nového protokolu?*

Tradiční TCP

- řízení toku (flow control) vs. řízení zahlcení (congestion control)



Tradiční TCP

- Řízení toku
  - explicitní zpětná vazba od příjemce pomocí *rwnd*
  - deterministické
- Řízení zahlcení
  - přibližný odhad pomocí odesílatelem určovaného *cwnd*
- Finální výslední výstupní okno *ownd*

$$ownd = \min\{rwnd, cwnd\}$$

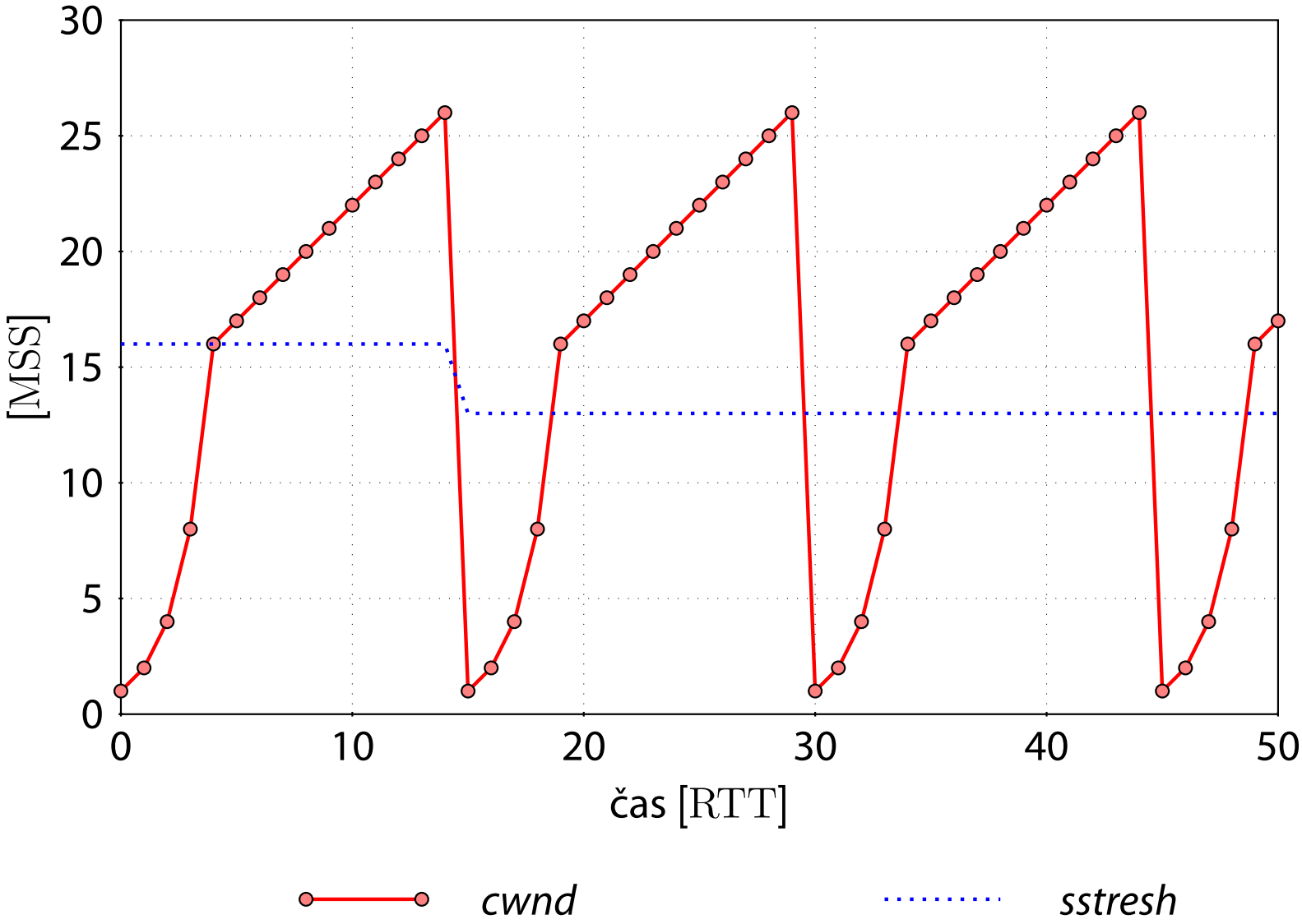
Použitá šířka pásma *bw* je pak

$$bw = \frac{8 \cdot ownd \cdot \mathrm{MTU}}{\mathrm{RTT}} \quad (1)$$

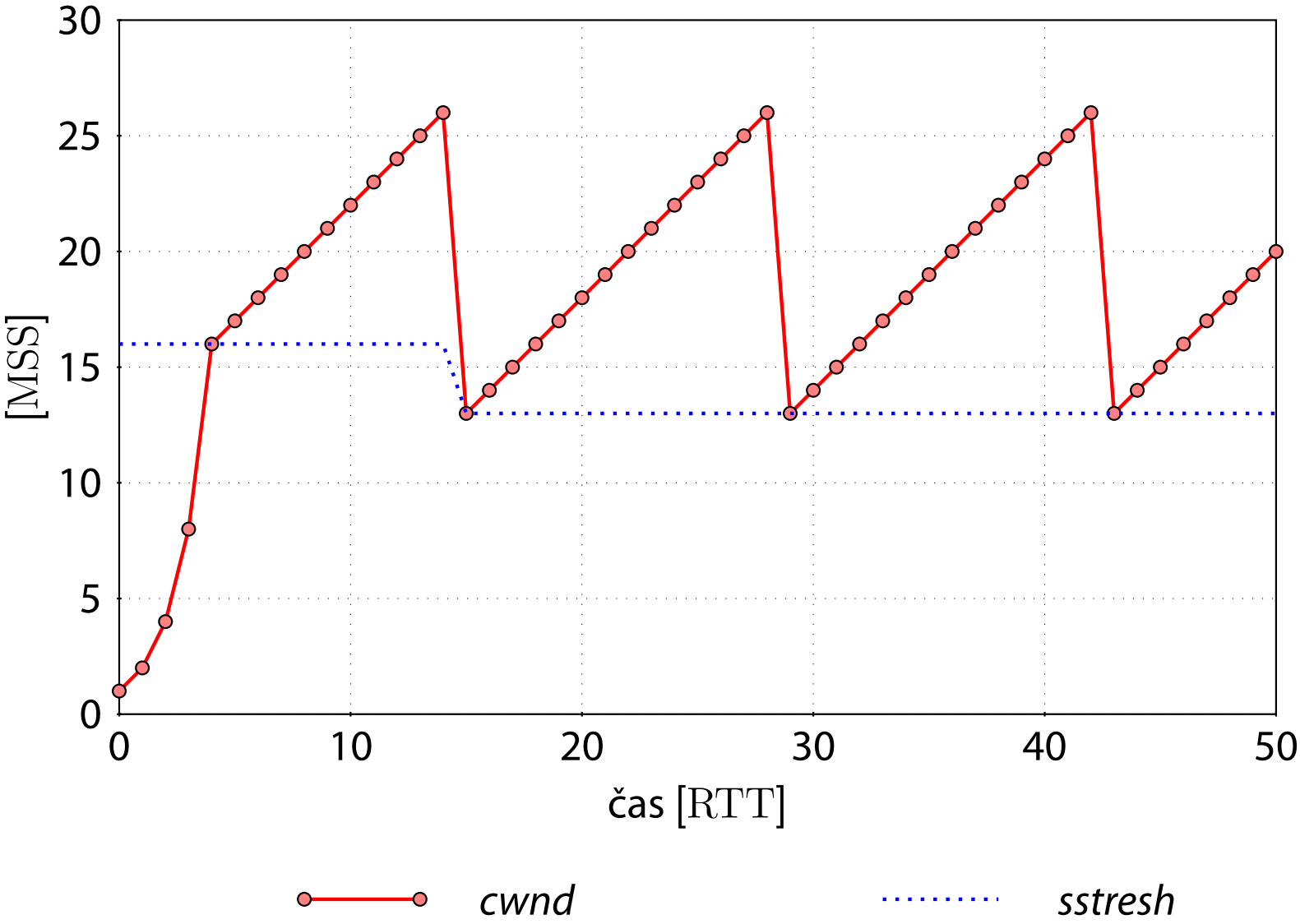
Tradiční TCP – Tahoe a Reno

- řízení zahlcení
  - tradičně založeno na přístupu *AIMD – Additive Increase Multiplicative Decrease*
  - Tahoe [1]
    - $cwnd = cwnd + \mathrm{MSS}$
      …za každý RTT bez výpadku nad hranicí *sstresh*
    - $sstresh = 0{,}5 cwnd$
      $cwnd = \mathrm{MSS}$
      …pro každý výpadek
  - Reno [2] přidává
    - *fast retransmission* (rychlé přeposlání) – ztráta indikovaná třemi po sobě jdoucími identickými ACKy
    - *fast recovery* (rychlá obnova) – zrušení slow-start fáze
      $sstresh = cwnd = 0{,}5 cwnd$

Tradiční TCP – Tahoe



Tradiční TCP – Reno



TCP Vegas

- Koncept řízení zahlcení Vegas [3]
  - při zahlcení sítě se začíná prodlužovat RTT
  - monitoring RTT v průběhu spojení
  - lineární zmenšování okna jako reakce na prodlužování RTT
- Možnost měření dostupného pásma měřením mezipaketové disperze (inter-packet spacing/dispersion)

Tradiční TCP

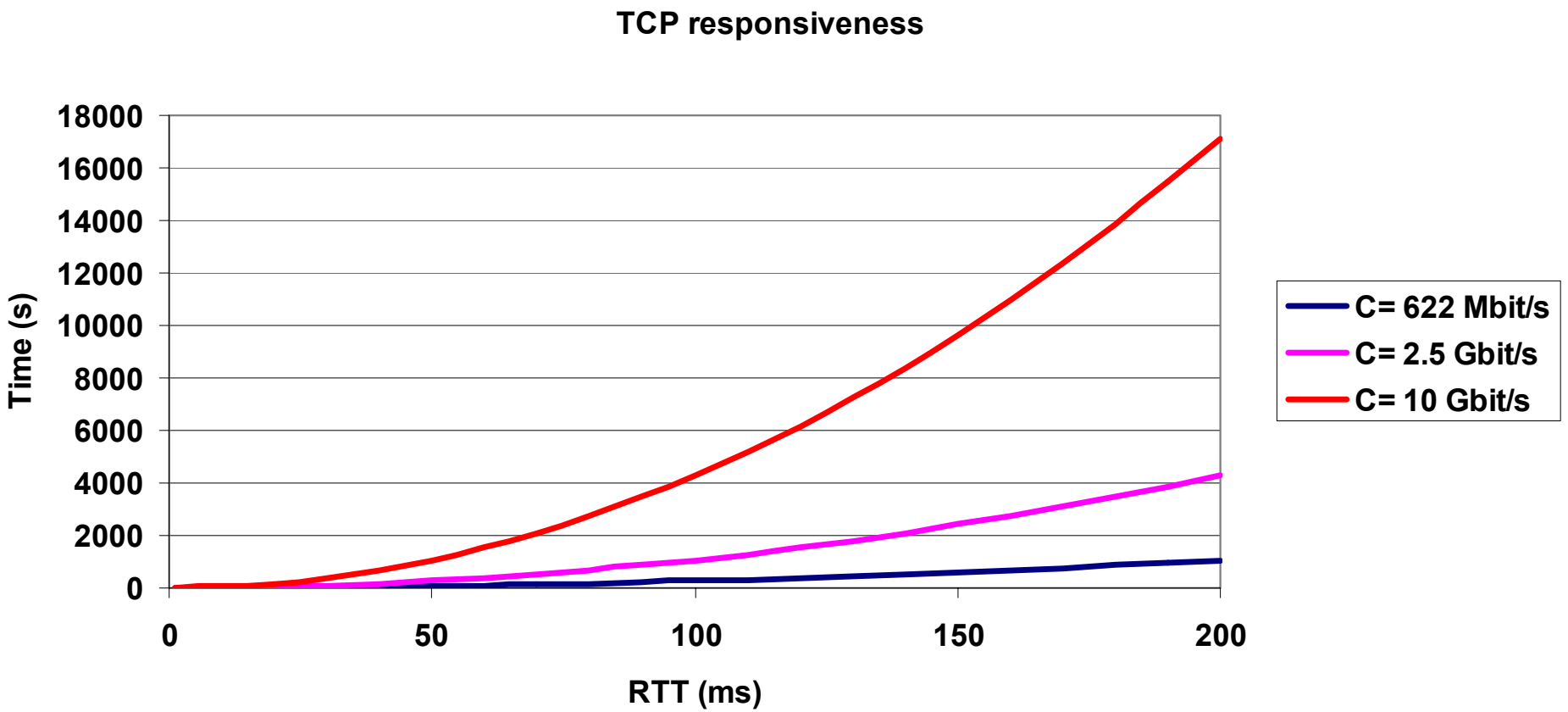
- Reakce na ztrátu dat – přeposlání
  - Tahoe: celé současné okno *ownd*
  - Reno: jeden segment v režimu Fast Retransmission
  - NewReno: více segmentů v režimu Fast Retransmission
  - Selective Acknowledgement (SACK): pouze ztracené pakety
- Základní otázka:
  Jak dosáhnout za realistických podmínek dostatečně velké cwnd na síti s velkým součinem kapacity a RTT?
  …a jak přitom neznemožnit přístup k síťové kapacitě pro „běžné“ uživatele?

Tradiční TCP – Response Function

- Response Function vyjadřuje vztah mezi *bw* a rovnovážnou frekvencí výpadků paketů *p* (steady-state packet loss rate)
  - $ownd \approx \frac{1{,}2}{\sqrt{p}}$
  - dosazením z (1) $bw \approx \frac{9{,}6\,\mathrm{MSS}}{\mathrm{RTT}\sqrt{p}}$
- Resposivnost tradičního TCP
  - za předpokladu, že k výpadku došlo když $cwnd = bw \cdot \mathrm{RTT}$

$$\varrho = \frac{bw\,\mathrm{RTT}^2}{2\mathrm{MSS}}$$

Tradiční TCP – Responsivnost

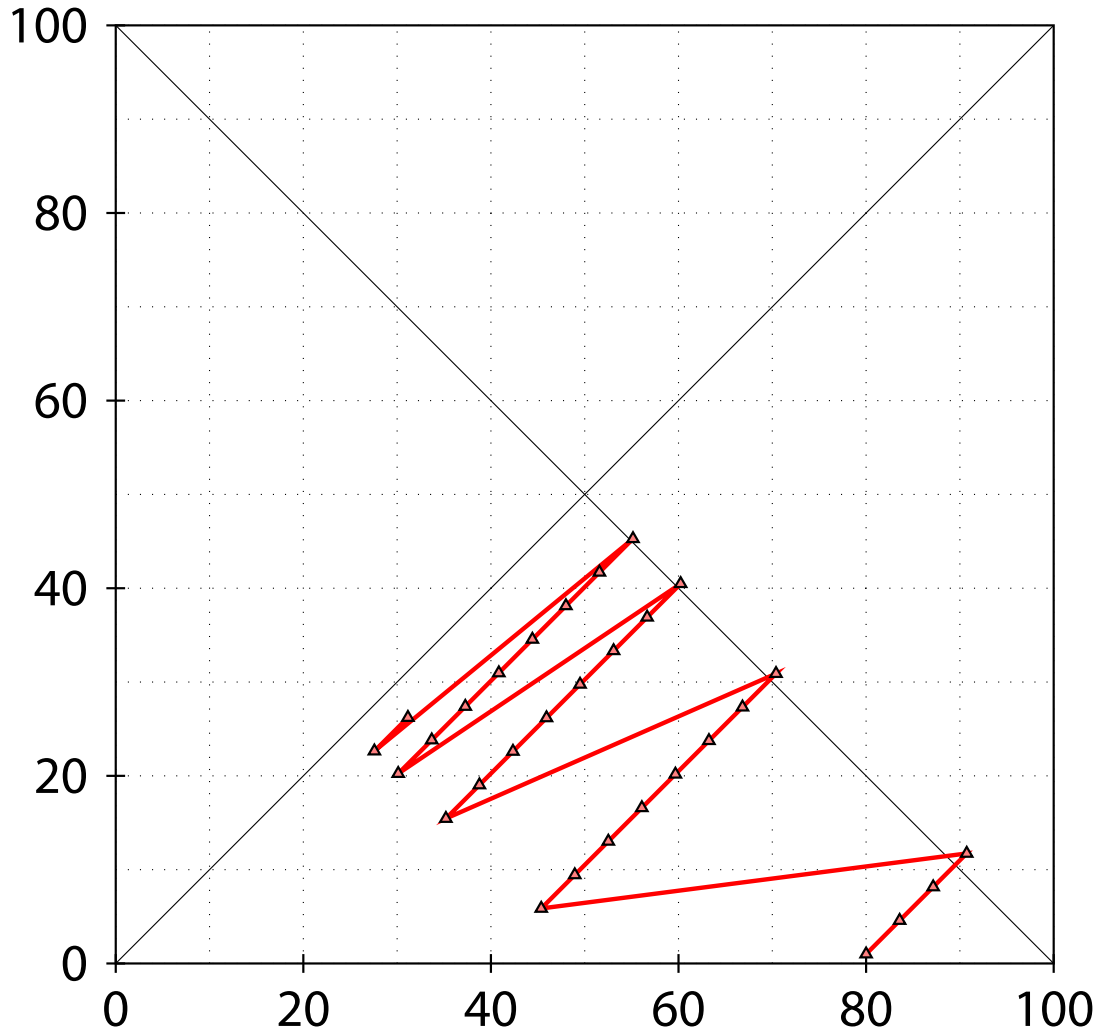


Tradiční TCP – Férovost

- Férovost v rovnovážném stavu
- Posuzování férovosti
  - pro proudy s různou RTT
  - pro proudy s různou MTU
- Podstatná je také rychlost konvergence do rovnovážného stavu!

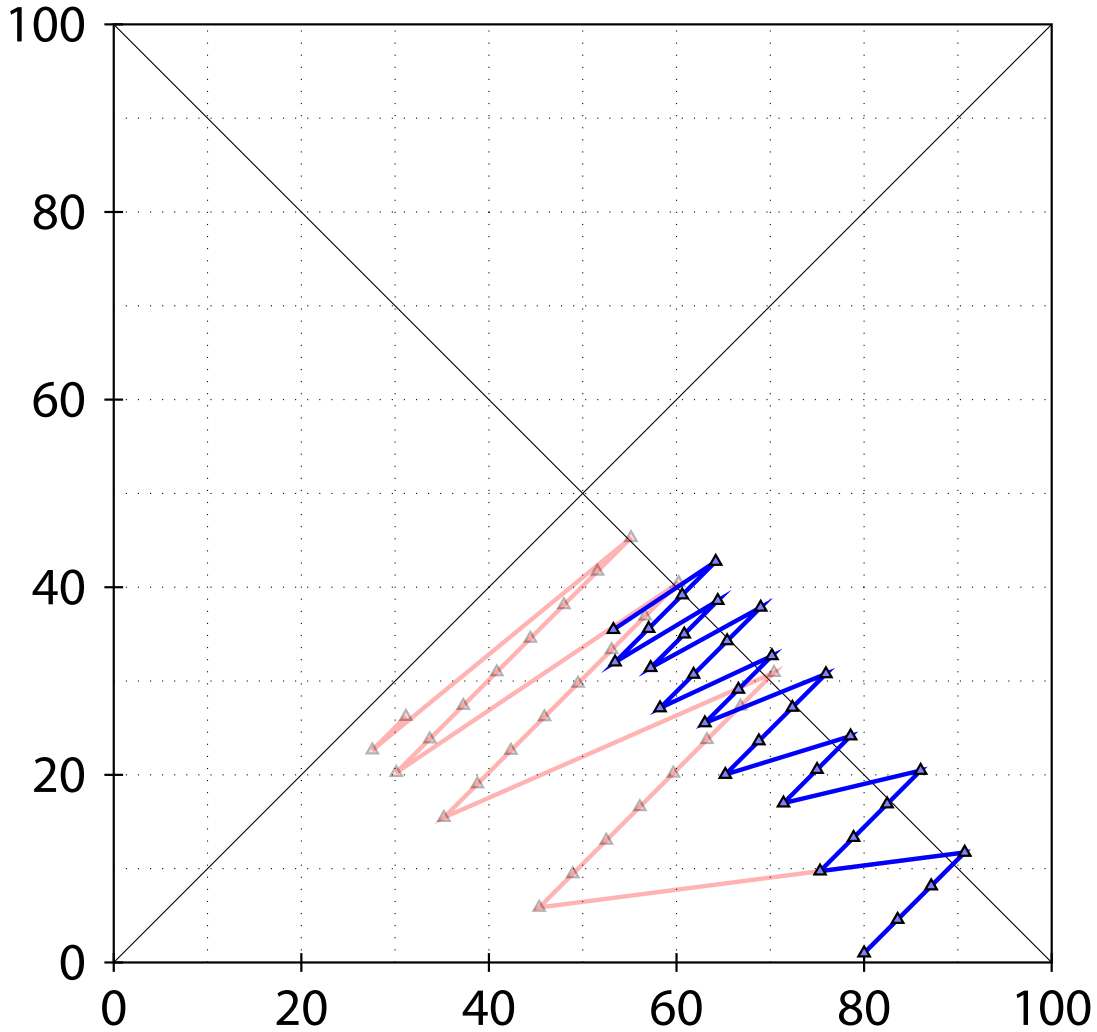
Tradiční TCP – Férovost

- $cwnd \mathrel{+}= \mathrm{MSS}$, $cwnd \mathrel{*}= 0{,}5$



Tradiční TCP – Férovost

- $cwnd \mathrel{+}= \mathrm{MSS}$, $cwnd \mathrel{*}= 0{,}83$



Přehled přednášky

Víceproudové TCP

- Zlepšuje chování TCP prakticky pouze v případě, že nastávají izolované výpadky paketů
- Výpadek více paketů obvykle ovlivní více proudů
- Často dostupné díky snadné implementaci
  - bbftp, GridFTP, Internet Backplane Protocol, …
- Nevýhody:
  - komplikovanější než TCP (obvykle více vláken)
  - nastartování je zrychleno nanejvýš lineárně
  - *synchronní přetěžování front a vyrovnávacích pamětí na směrovačích*

Ladění implementace TCP

- Spolupráce s HW
  - Rx/Tx TCP Checksum Offloading
  - běžně dostupné (ale někdy obsahuje chyby)
- Zero copy
  - přístup k síti obvykle zahrnuje několik kopií dat: user-land $\leftrightarrow$ kernel $\leftrightarrow$ síťová karta
  - page flipping – přesun user-land $\leftrightarrow$ kernel
    - podpora např. pro `sendfile()`
  - implementace pro Linux, FreeBSD, Solaris, …

Ladění implementace TCP

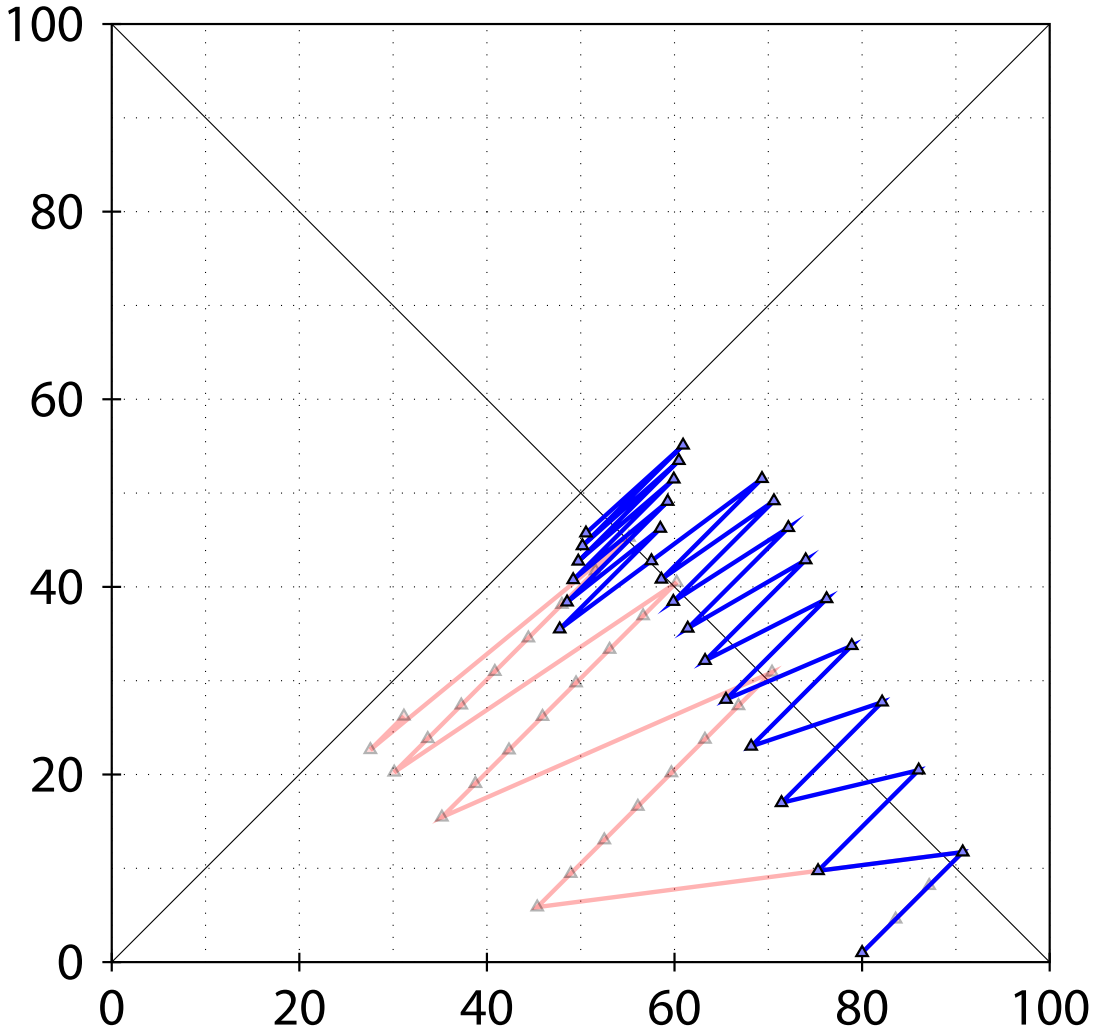
- Web100 [4, 5]
  - instrumentace TCP/IP stacku pro Linux – TCP Kernel Instrumentation Set (TCP-KIS)
    - více jak 125 „táhel“
    - informace jsou dostupné přes `/proc`
  - knihovna pro přístup k instrumentaci
  - klientské nástroje v uživatelském prostoru (command-line, GUI)
  - monitoring
  - ladění parametrů
  - podpora pro auto-tuning

Přehled přednášky

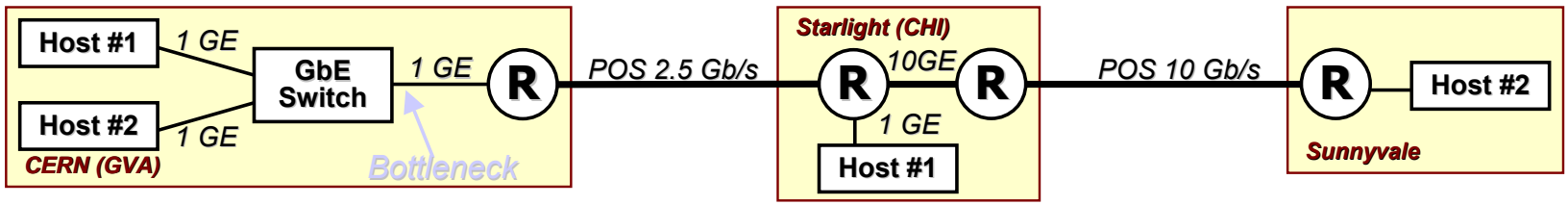
GridDT

- sbírka ad-hoc modifikací :(
- korekce *sstresh*
  - rychlejší slowstart
- modifikace AIMD řízení zahlcení
  - $cwnd = cwnd + a$
    …pro úspěšné RTT
  - $cwnd = b\,cwnd$
    …pro výpadek
- modifikace pouze na straně odesílače

GridDT – férovost



GridDT – příklad



TCP Reno performance (see slide #8):

First stream GVA <-> Sunnyvale : RTT = 181 ms ; Avg. throughput over a period of 7000s = 202 Mb/s

Second stream GVA<->CHI : RTT = 117 ms; Avg. throughput over a period of 7000s = 514 Mb/s

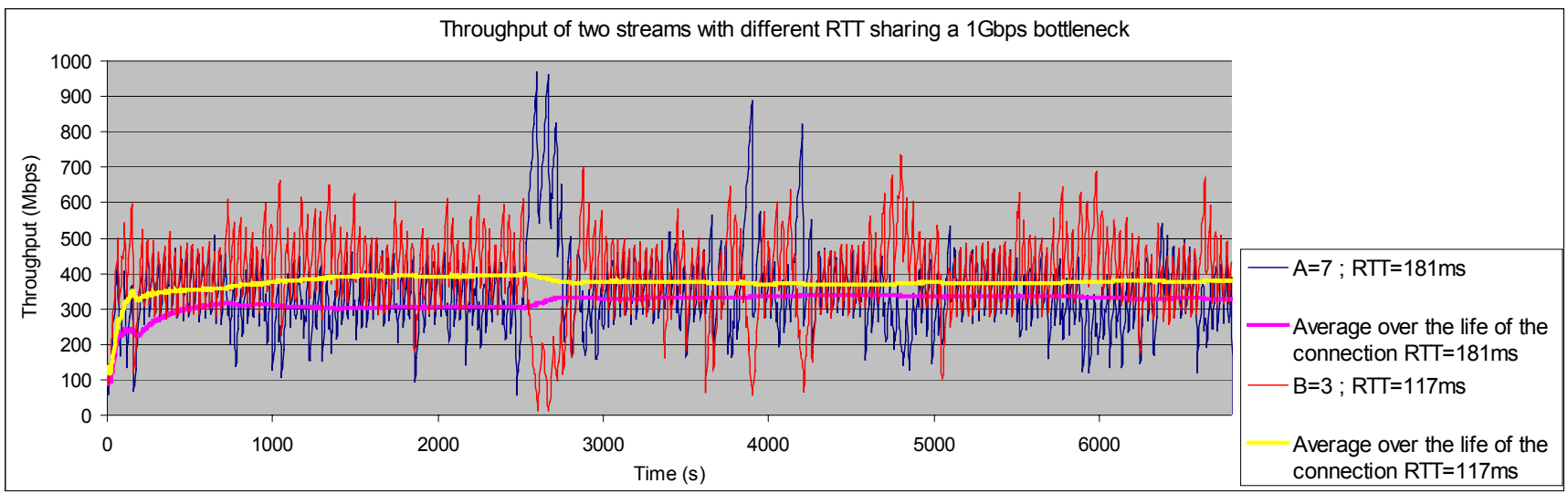
Links utilization 71,6%

Grid DT tuning in order to improve fairness between two TCP streams with different RTT:

First stream GVA <-> Sunnyvale : RTT = 181 ms, Additive increment = A = 7 ; Average throughput = 330 Mb/s

Second stream GVA<->CHI : RTT = 117 ms, Additive increment = B = 3 ; Average throughput = 388 Mb/s

Links utilization 71.8%



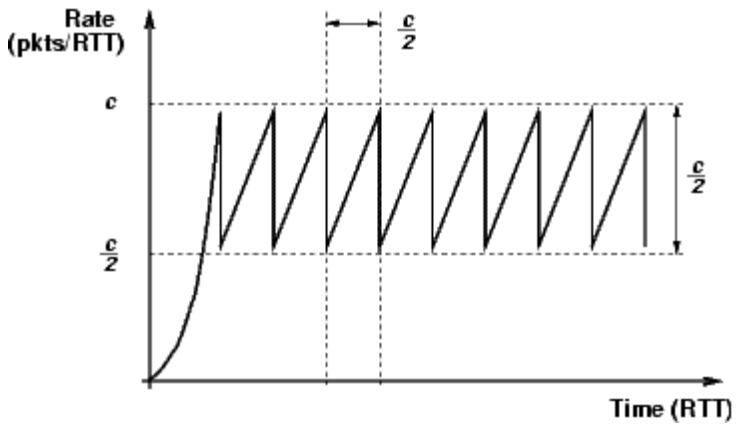
Scalable TCP

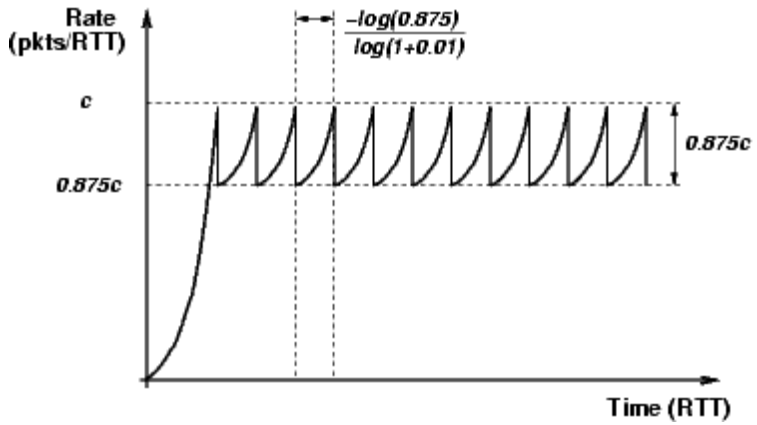
- navrhl Tom Kelly [1]
- řízení zahlcení již není AIMD:
  - $cwnd = cwnd + 0{,}01\, cwnd$
    …pro úspěšné RTT
    $cwnd = cwnd + 0{,}01$
    …per-ACK
  - $cwnd = 0{,}875\, cwnd$
    …pro výpadek

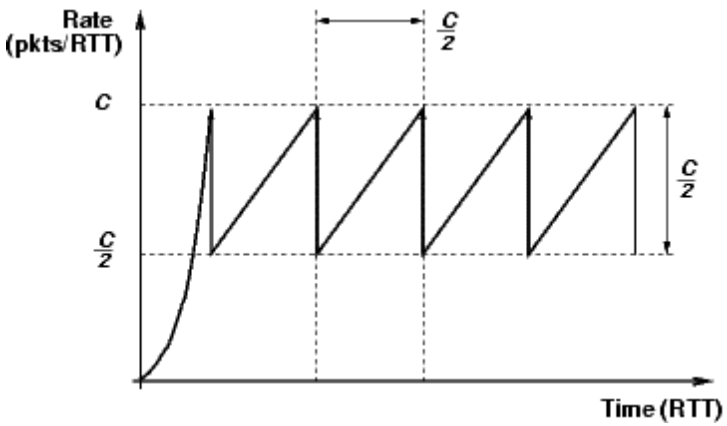
$\Longrightarrow$ Multiplicative Increase Multiplicative Decrease (MIMD)

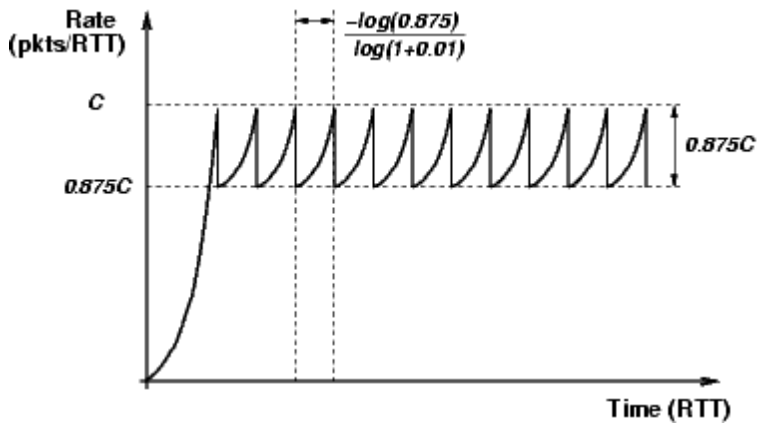
  - pro malé velikosti okna a/nebo větší množství ztrát v síti se přepíná od AIMD režimu

Scalable TCP



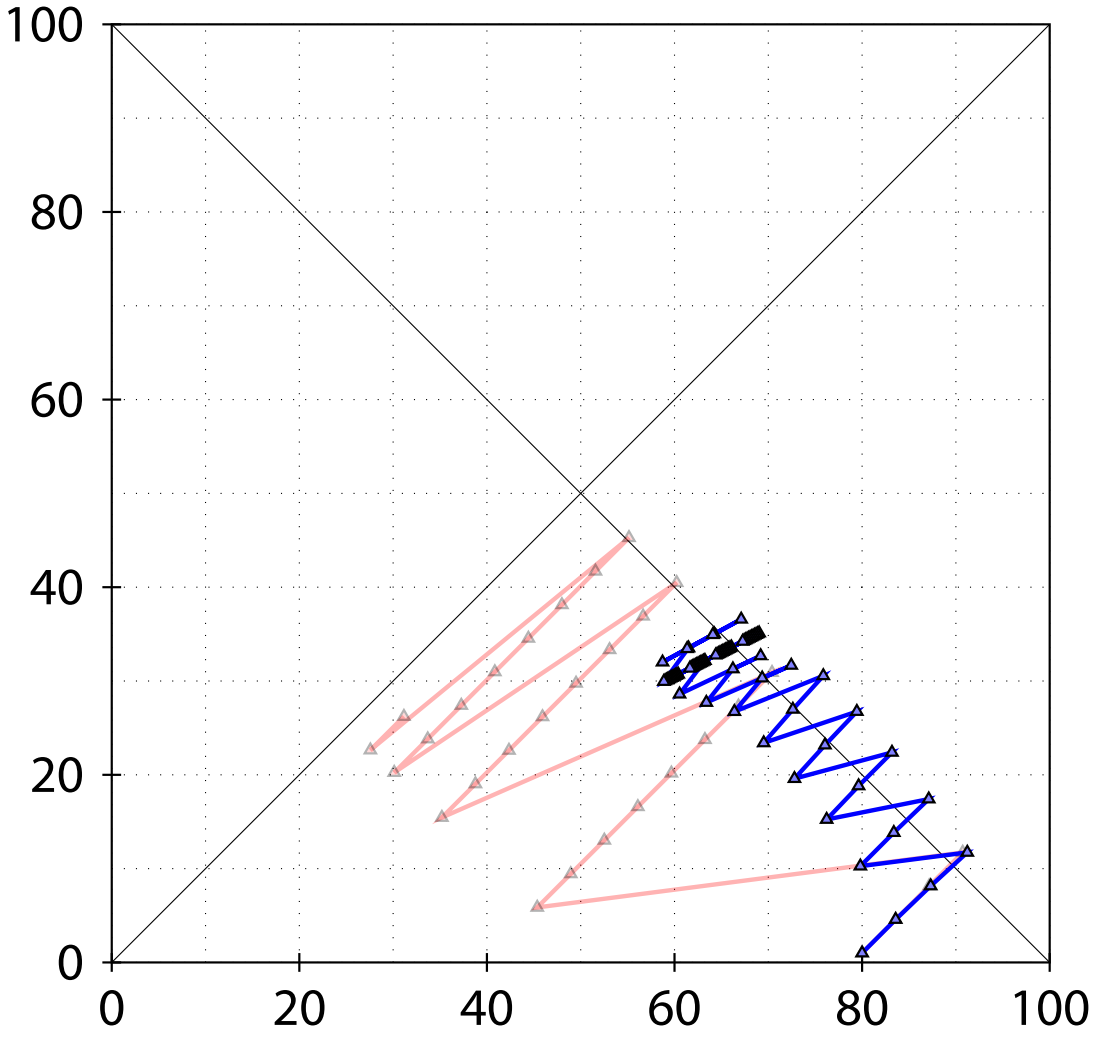






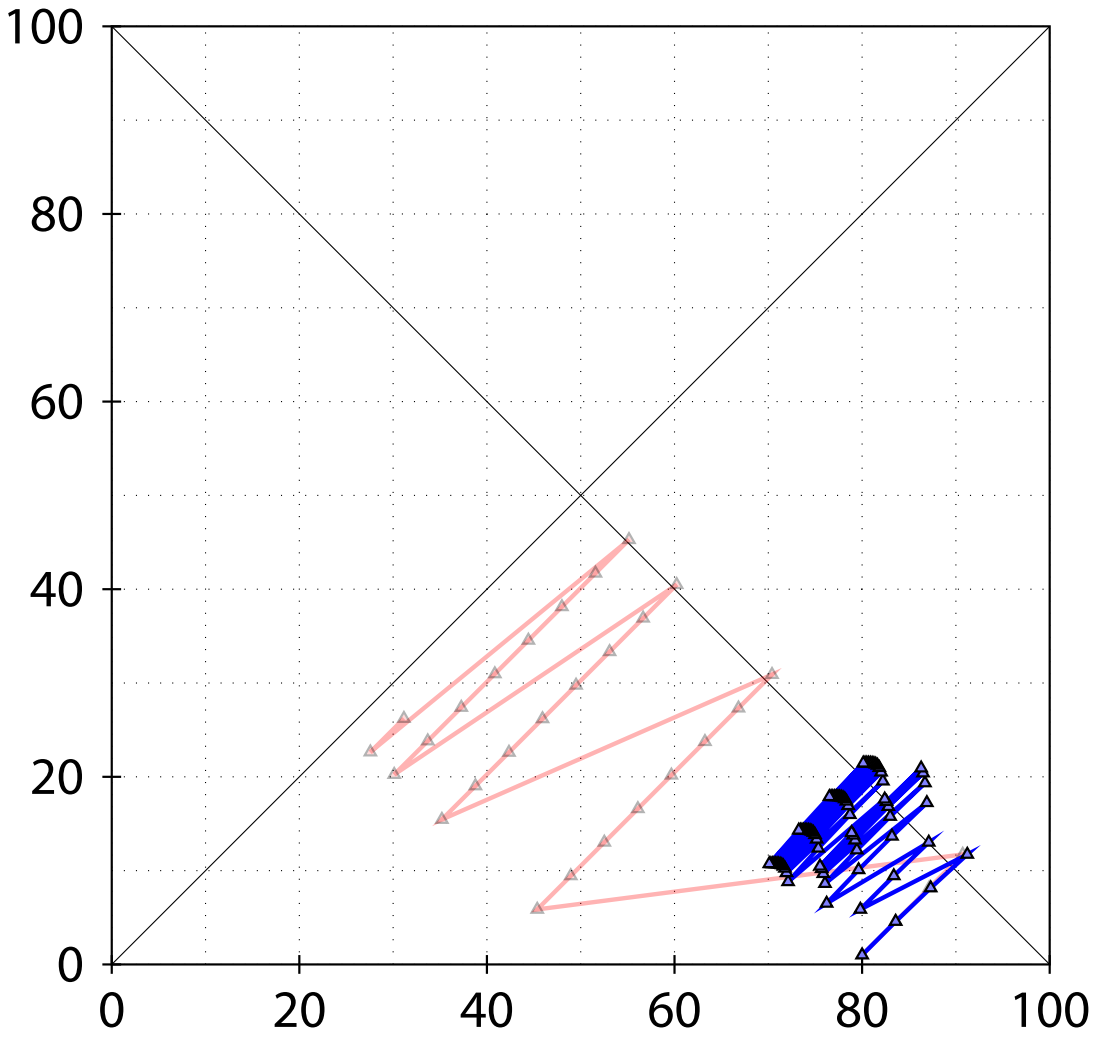
Scalable TCP – férovost

dva soutěžící Scalable TCP proudy, přepnutí na Scalable řízení @ $>$30Mb/s, dvojnásobek kroků oproti předchozím simulacím

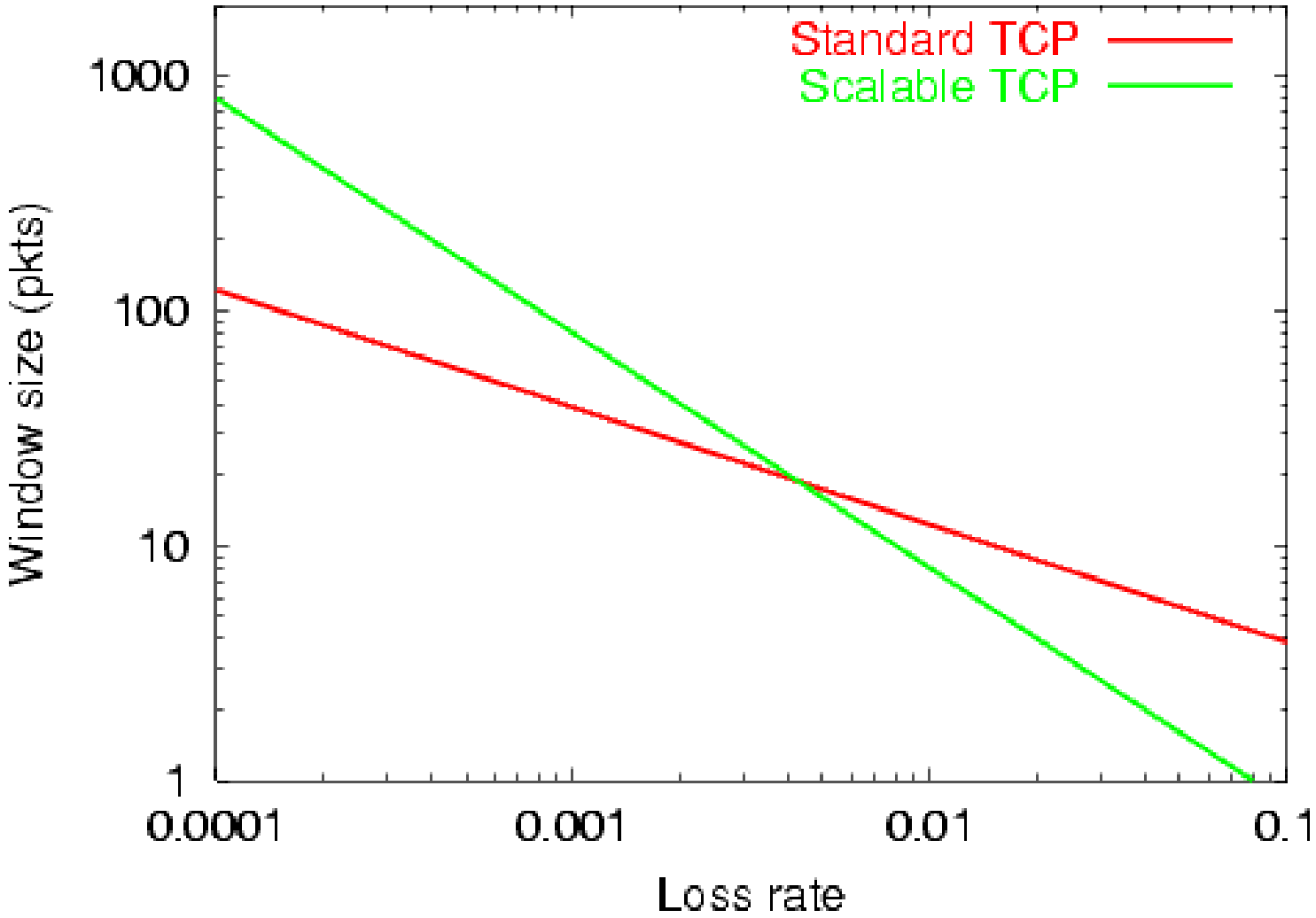


Scalable TCP – férovost

soutěžící Scalable TCP a tradiční TCP proudy, přepnutí na Scalable řízení @ >30Mb/s, dvojnásobek kroků



Scalable TCP – response curve



High-Speed TCP (HSTCP)

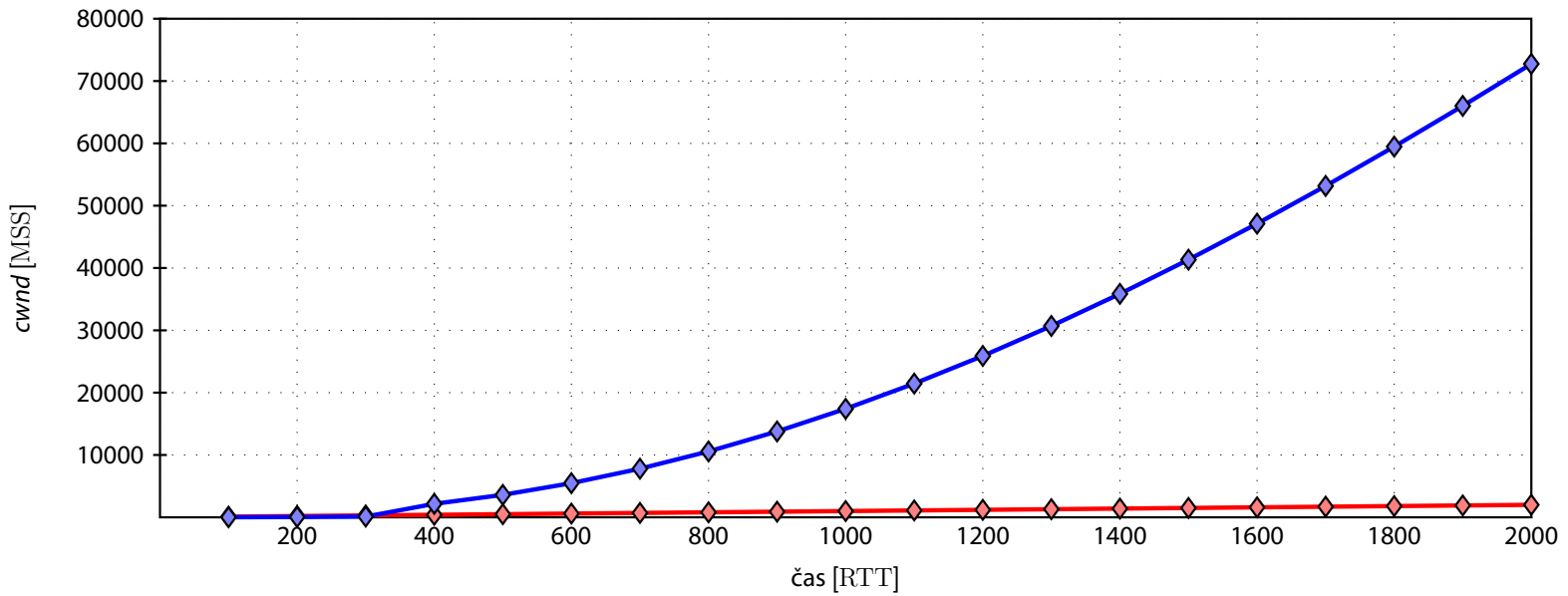
- Sally Floyd, RFC3649, [2]
- řízení zahlcení AIMD/MIMD:
  - $cwnd = cwnd + a(cwnd)$
    …pro úspěšné RTT
    $cwnd = cwnd + \frac{a(cwnd)}{cwnd}$
    …per-ACK
  - $cwnd = b(cwnd)\, cwnd$
    …pro výpadek
- emuluje chování tradičního TCP pro malé velikosti okna a/nebo větší množství ztrát v síti

High-Speed TCP (HSTCP)

- navržená parametrizace MIMD:

$$b(cwnd) = \frac{-0{,}4(\log(cwnd) - 3{,}64)}{7{,}69} + 0{,}5$$

$$a(cwnd) = \frac{2cwnd^2 b(cwnd)}{12{,}8(2 - b(cwnd))w^{1,2}}$$



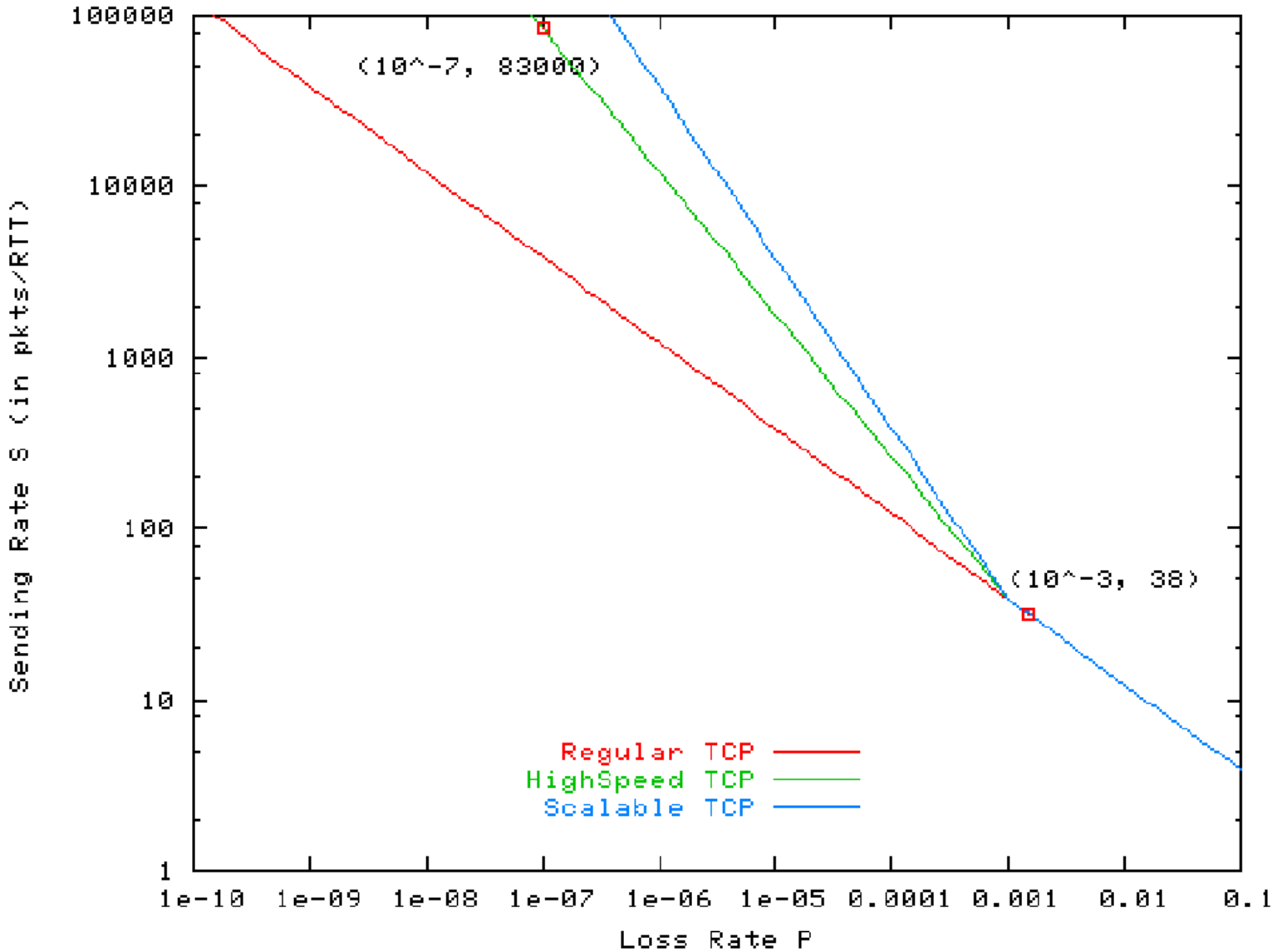
High-Speed TCP (HSTCP)

- možná parametrizace ekvivalentní Scalable TCP: Linear HSTCP
- porovnání s víceproudovým TCP

$$N(cwnd) \approx 0{,}23 cwnd^{0,4}$$

- ani Scalable TCP ani HSTCP neřeší nijak sofistikovaně fázi slow-start

HSTCP – response curve



H-TCP

- Δ … čas uplynulý od minulého výpadku
- přírůstek *cwnd* závisí na Δ jakožto indikaci (šířka pásma $\times$ zpoždění) a také na RTT, aby se kompenzovala neférovost mezi toky s různým RTT
- Δ_L … pro $\Delta \leq \Delta_L$ se používá TCP nárůst
- Δ_B … hranice změny dostupné šířky pásma, nad níž se používá TCP pokles (pro velké změny dostupné šířky pásma se používá TCP pokles 0,5)
- T_{min}, T_{max} … minimální resp. maximální změřené RTT
- $B(k+1)$ … měření maximální propustnosti za poslední interval bez výpadku

H-TCP

- $cwnd = cwnd + \frac{2(1-\beta)\,a(\Delta)}{cwnd}$
  …per-ACK
- $cwnd = b(B)\,cwnd$
  …pro výpadek

$$a(\Delta) = \begin{cases} 1 & \Delta \leq \Delta_L \\ \max\{a'(\Delta)T_{min}; 1\} & \Delta > \Delta_L \end{cases}$$

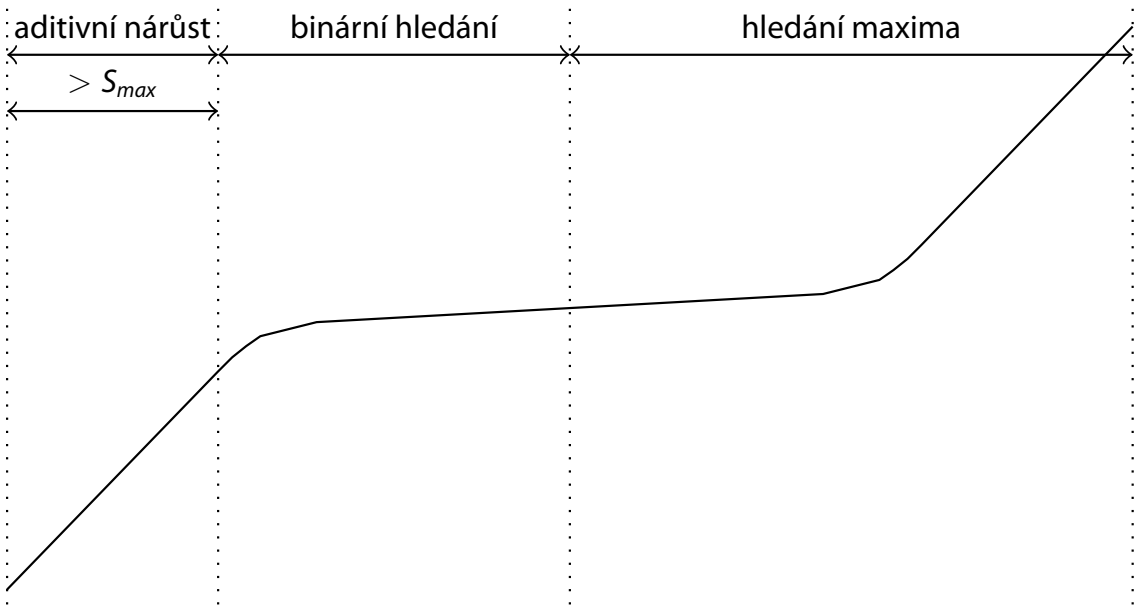
$$b(B) = \begin{cases} 0{,}5 & \left|\frac{B(k+1)-B(k)}{B(k)}\right| > \Delta_B \\ \min\{\frac{T_{min}}{T_{max}}; 0{,}8\} & \text{v opačném případě} \end{cases}$$

$$a'(\Delta) = 1 + 10(\Delta - \Delta_L) + 0{,}25(\Delta - \Delta_L)^2$$

…kvadratická přírůstková funkce

BIC-TCP

- K aktualizaci *cwnd* používá binární prohledávací algoritmus [3]
- 4 fáze fungování
  (1) reakce na výpadek
  (2) aditivní nárůst
  (3) binární prohledávání
  (4) hledání maxima



BIC-TCP

(1) Výpadek
- redukce okna
- původní okno $\longrightarrow W_{max}$
- redukované okno $\longrightarrow W_{min}$
- $\Longrightarrow$ protože k výpadku došlo při $cwnd \leq W_{max}$, budeme rovnovážné $cwnd$ hledat v intervalu $\langle W_{min}; W_{max} \rangle$

(2) Aditivní nárůst
- začít hledání od $cwnd = \frac{W_{min}+W_{max}}{2}$ by mohlo být pro síť příliš náročné
- pokud $\frac{W_{min}+W_{max}}{2} > W_{min} + S_{max}$, postupujeme aditivním nárůstem o konstantu $cwnd = W_{min} + S_{max}$

BIC-TCP

(3) Binární hledání
- $cwnd = \frac{W_{min}+W_{max}}{2}$
- pokud předchozí bod (i aditivní nárůst) prošel bez výpadku, $W_{min} = cwnd$, v opačném případě $W_{max} = cwnd$
- hledání pokračuje, pokud změna *cwnd* není menší než konstanta S_{min}, kdy se nastaví $cwnd = W_{max}$

- výsledkem bodů (2) a (3) je obvykle lineární růst (aditivní nárůst), který se mění na logaritmický (binární hledání)

BIC-TCP

(4) Hledání maxima
- inverzní proces k bodům (3) a (2)
- nejdříve inverzní binární hledání, dokud nárůst není větší jako S_{max}
- lineární nárůst o velký inkrement po překročení předchozího bodu

- očekáváné výhody
  - „přáteskost“ vůči TCP
    - během plata (3) mají TCP toky šanci „dorůst“
    - AIMD chování (byť rychlejší) ve fázích (2) a (4)
  - stabilnější velikost okna $\Longrightarrow$ lepší využití sítě
    - většinu času by BIC-TCP mělo trávit v platu (3)

BIC-TCP versus CUBIC

- BIC-TCP v. 2.0 se označuje jako CUBIC

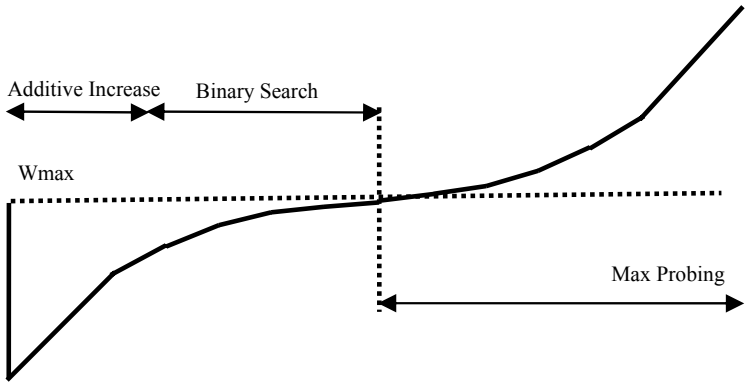


Fig. 1: The Window Growth Function of BIC

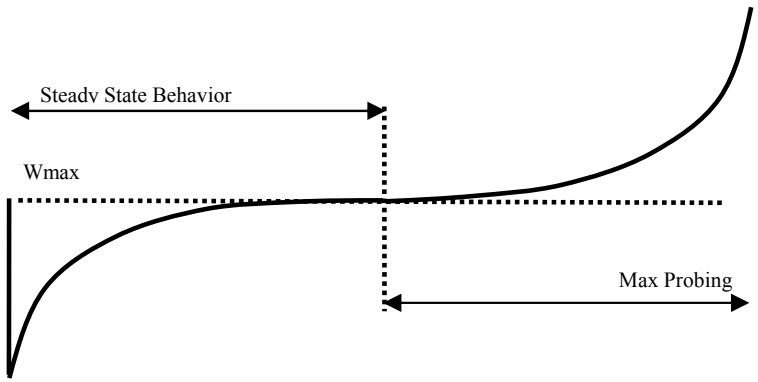


Fig. 2: The Window Growth Function of CUBIC

`http://www4.ncsu.edu/~rhee/export/bitcp/cubic-paper.pdf`

Srovnání variant

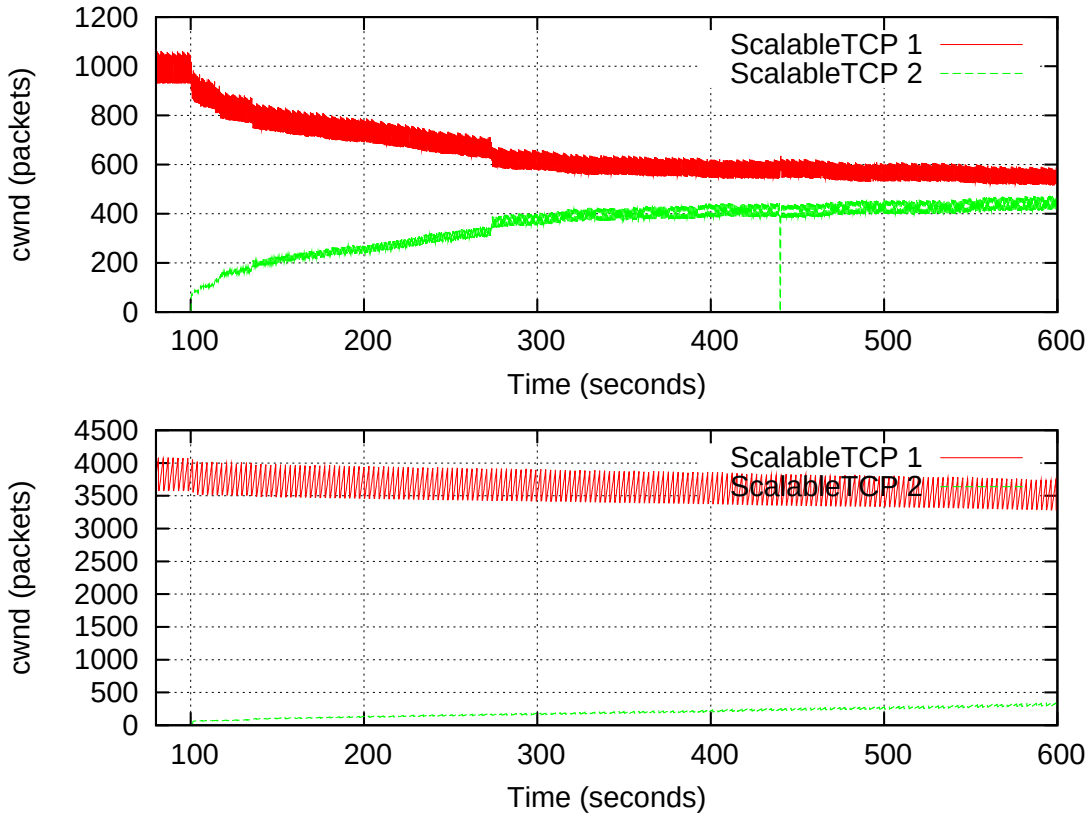


Fig. 4. Scalable-TCP *cwnd* time histories following startup of a second flow. RTT of both flows is 42ms (top) and 162ms (bottom). Bottleneck bandwidth is 250Mbit/sec, queue size 20% BDP, no web traffic.

http://www.hamilton.ie/net/eval/ToNfinal.pdf

Srovnání variant

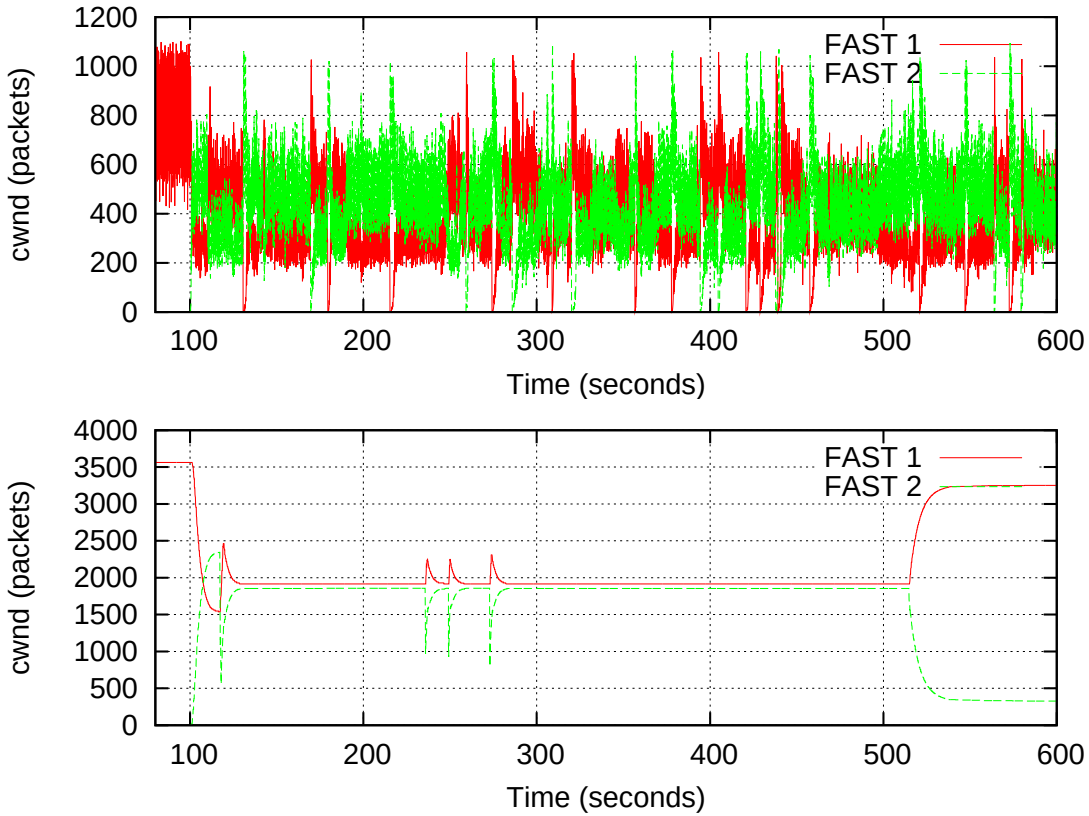


Fig. 5. FAST-TCP *cwnd* time histories following startup of a second flow. RTT is 42ms (top) and 162ms (bottom). Bottleneck bandwidth is 250Mbit/sec, queue size 20% BDP, no web traffic.

http://www.hamilton.ie/net/eval/ToNfinal.pdf

Srovnání variant

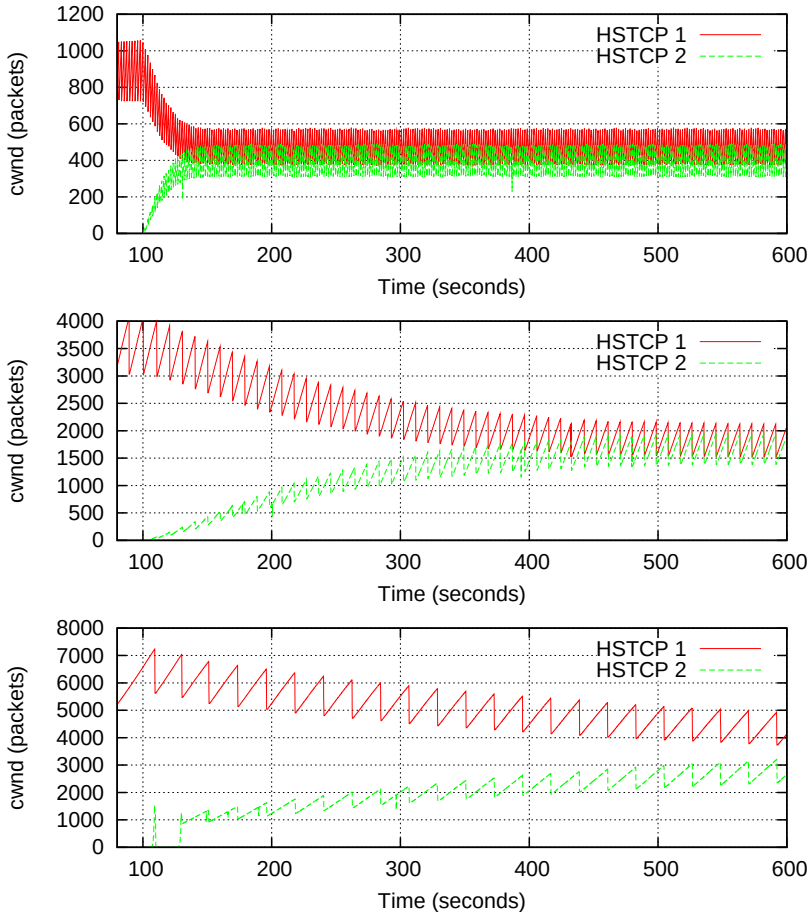


Fig. 6. HS-TCP *cwnd* time histories following startup of a second flow. RTT is 42ms (top), 162ms(middle) and 324ms (bottom). Bottleneck bandwidth 250Mbit/sec, queue size 20% BDP, no web traffic.

http://www.hamilton.ie/net/eval/ToNfinal.pdf

Srovnání variant

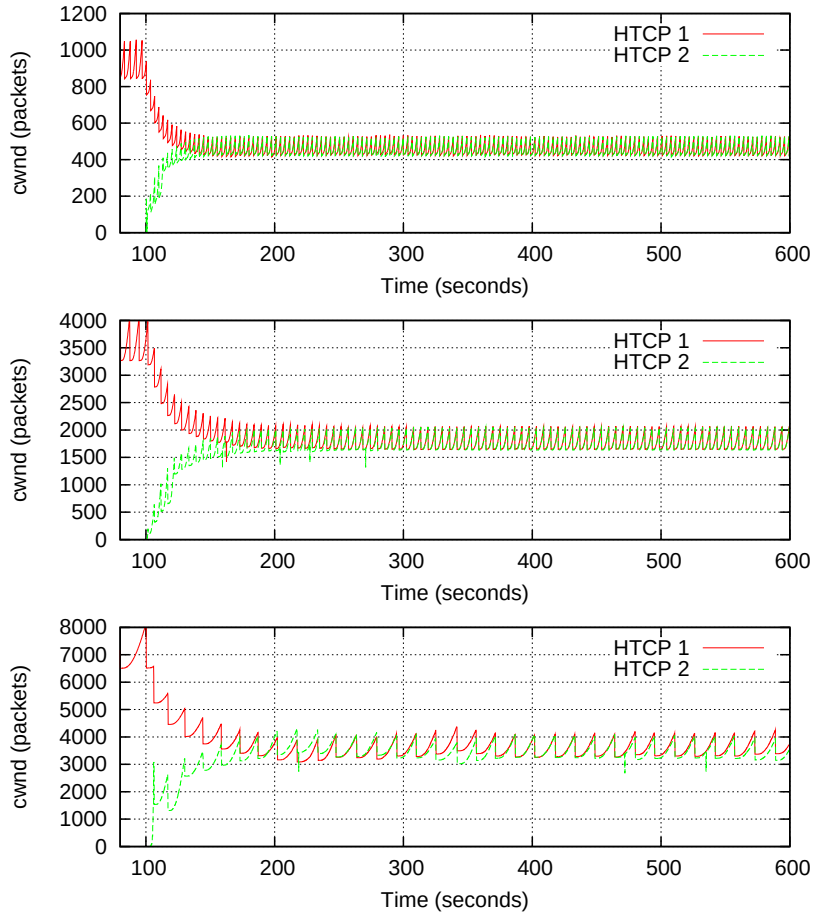


Fig. 9. H-TCP *cwnd* time histories following startup of a second flow. RTT is 42ms (top), 162ms (middle) and 324ms (bottom). Bottleneck bandwidth is 250Mbit/sec, queue size 20% BDP, no web traffic.

http://www.hamilton.ie/net/eval/ToNfinal.pdf

Srovnání variant

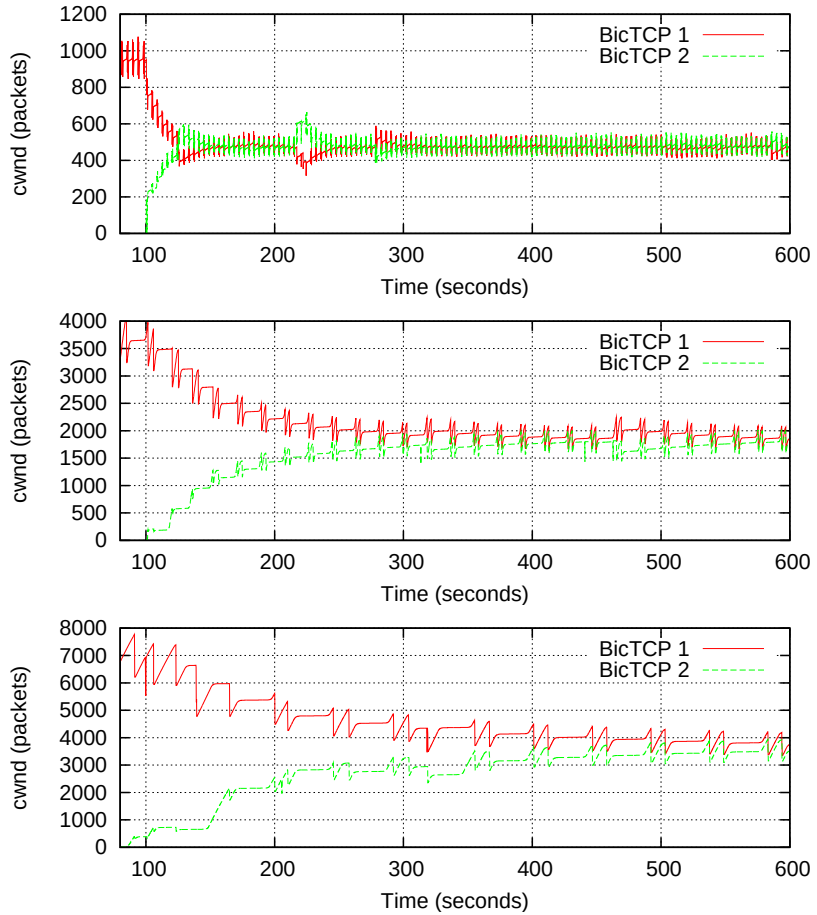


Fig. 8. BIC-TCP *cwnd* time histories following startup of a second flow. RTT is 42ms (top), 162ms(middle) and 324ms (bottom). Bottleneck bandwidth is 250Mbit/sec, queue size 20% BDP, no web traffic.

http://www.hamilton.ie/net/eval/ToNfinal.pdf

Quickstart (QS)/Limited Slowstart

- existuje silné podezření, že slow-start fáze se nedá vylepšit bez interakce s níže položenými síťovými vrstvami
- návrh: 4-byte option v IP hlavičce, který zahrnuje pole $\mathrm{QS\ TTL}$ a $\mathrm{Initial\ Rate}$
- odesílač, který chce použít QS, nastaví $\mathrm{QS\ TTL}$ na náhodnou hodnotu a $\mathrm{Initial\ Rate}$ na požadovanou rychlost, kterou chce začít vysílat, a pošle SYN paket

Quickstart (QS)/Limited Slowstart

- všechny routery po cestě, které podporují QS, sníží $\mathrm{QS\ TTL}$ o jedničku a sníží $\mathrm{Initial\ Rate}$, pokud je to potřeba
- Přijímač pošle pole $\mathrm{QS\ TTL}$ a $\mathrm{Initial\ Rate}$ v SYN/ACK paketu odesílači
- Odesílač ví, jestli všechny směrovače po cestě podporují QS (porovnáním $\mathrm{QS\ TTL}$ a TTL)
- Odesílač si nastaví příslušné *cwnd* a začne používat svůj mechanismus řízení zahlcení (např. AIMD)
- Vyžaduje změny v IP vrstvě! :-(

E-TCP

- Early Congestion Notification (ECN)
  - součást Advanced Queue Management (AQM)
  - bit, který nastavují routery, když se blíží linky/buffery/fronty zahlcení
  - ECN příznak musí být odzrcadlen přijímačem
  - na ECN bit má TCP zareagovat stejně jako na výpadek
  - problém s tím, aby správci routerů konfigurovali AQM/ECN :-(

E-TCP

- E-TCP
  - navrhuje odzrcadlit ECN bit jen jednou (poprvé)
  - zamrzne *cwnd* když dorazí od přijímače ACK s nastaveným ECN bitem
  - vyžaduje (umělé) zavedení malých náhodných výpadků do sítě, aby se zajistil multiplikativní pokles kvůli férovému chování v čase
  - vyžaduje změnu chování k ECN bitu na přijímačích :-(

FAST

- Fast AQM Scalable TCP (FAST) [5]
- používá end-to-end delay, ECN a ztráty paketů pro detekci/vyhýbání se zahlcení
- T_{min}, T' … minimální a průměrný pozorovaný RTT
- T_q … odhad zpoždění front u RTT
- $f'_\alpha(B)$ … (8, 20, 200) pro *bw* ($<$ 10 Mb/s, 10 $-$ 100 Mb/s, $>$ 100 Mb/s), lze měnit přes `sysctl()`
- γ … parametr návrhu ;-)

$$\text{ACK: } cwnd = \min\left\{2 \times cwnd, (1-\gamma)cwnd + \gamma\left[\frac{T_{min}}{T'}cwnd + f_\alpha(B, T_q)\right]\right\}$$

$$\text{výpadek: } cwnd = 0{,}5\, cwnd$$

$$f_\alpha(B, T_q) = \begin{cases} a \times cwnd & T_q = 0 \\ f'_\alpha(B) & T_q \neq 0 \end{cases}$$

Algoritmy v jádrech OS

- Linux
  - BIC-TCP (default 2.6.8–2.6.18)
  - CUBIC (default 2.6.19–)
  - dostupné implementace ve 2.6.22:
    Reno, Vegas, BIC, CUBIC, Westwood, H- TCP, High speed, Hybla, Scalable, Yeah, VENO, Illinois, and Low Priority
  - `http://git.kernel.org/?p=linux/kernel/git/torvalds/linux.git;a=blob_plain;f=net/ipv4/Kconfig;hb=HEAD`

    `TCP_CONG_ADVANCED`
  - TCP Implementation in Linux: A Brief Tutorial, `http://www.ece.virginia.edu/cheetah/documents/papers/TCPlinux.pdf`

Algoritmy v jádrech OS

- Linux

```
hopet@frakira:/lib/modules/2.6.24-24-server$ find . -name "*tcp*" \
| fgrep ipv4
./kernel/net/ipv4/tcp_yeah.ko
./kernel/net/ipv4/tcp_vegas.ko
./kernel/net/ipv4/tcp_highspeed.ko
./kernel/net/ipv4/tcp_cubic.ko
./kernel/net/ipv4/tcp_lp.ko
./kernel/net/ipv4/tcp_scalable.ko
./kernel/net/ipv4/tcp_probe.ko
./kernel/net/ipv4/tcp_illinois.ko
./kernel/net/ipv4/tcp_bic.ko
./kernel/net/ipv4/tcp_veno.ko
./kernel/net/ipv4/tcp_hybla.ko
./kernel/net/ipv4/tcp_westwood.ko
./kernel/net/ipv4/tcp_htcp.ko

sysctl net.ipv4.tcp_available_congestion_control
sysctl -w net.ipv4.tcp_congestion_control=cubic
```

Algoritmy v jádrech OS

- FreeBSD 9.x
  - ztráty: NewReno, CUBIC, HTCP
  - latence: Vegas, HD and CHD (odolnost proti výpadkům nezpůsobeným ucpáním)
  - `http://caia.swin.edu.au/freebsd/5cc/`
  - `http://freebsdfoundation.blogspot.com/2011/03/summary-of-five-new-tcp-congestion.html`
  - [3]
- Windows Vista/7/2008 Server
  - Compound TCP

```
netsh interface tcp set global congestionprovider=ctcp
netsh interface tcp set global congestionprovider=none
netsh interface tcp show global
```

Přehled přednášky

tsunami

- TCP spojení pro out-of-band řídící kanál
  - vyjednávání parametrů přenosu
  - požadavky na retransmisi – používá NACKy místo ACKů
  - vyjednávání ukončení přenosu
- UDP kanál pro přenos dat
  - řízení zahlcení MIMD
  - vysoce konfigurovatelné
    - parametry MIMD, nastavení prahu chyb, maximální velikost fronty pro retransmisi, interval zasílání požadavků na retransmisi

Reliable Blast UDP – RBUDP

- podobné jako tsunami – out-of-band TCP kanál pro řízení, UDP pro přenos
- vytvořeno pro přenosy z disku na disk, příp. takové, kde kompletní přenášená data lze udržet v paměti vysílače
- posílá data uživatelem definovanou rychlostí
  - `app_perf` (klon iperfu) pro odhad kapacity sítě a přijímače

Reliable Blast UDP – RBUDP

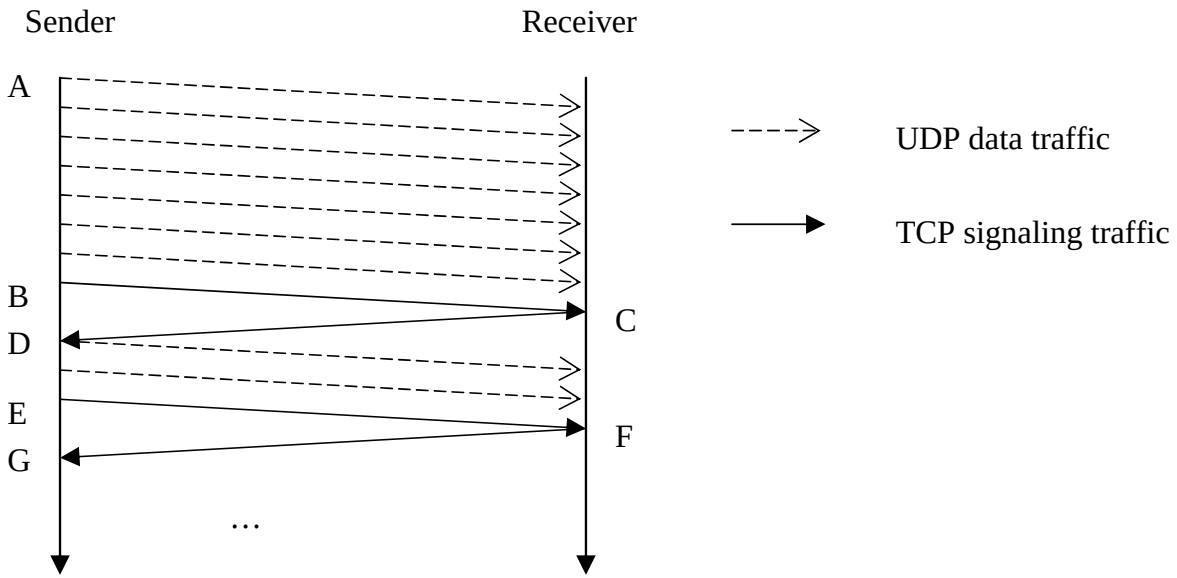


Figure 1. The Time Sequence Diagram of RBUDP

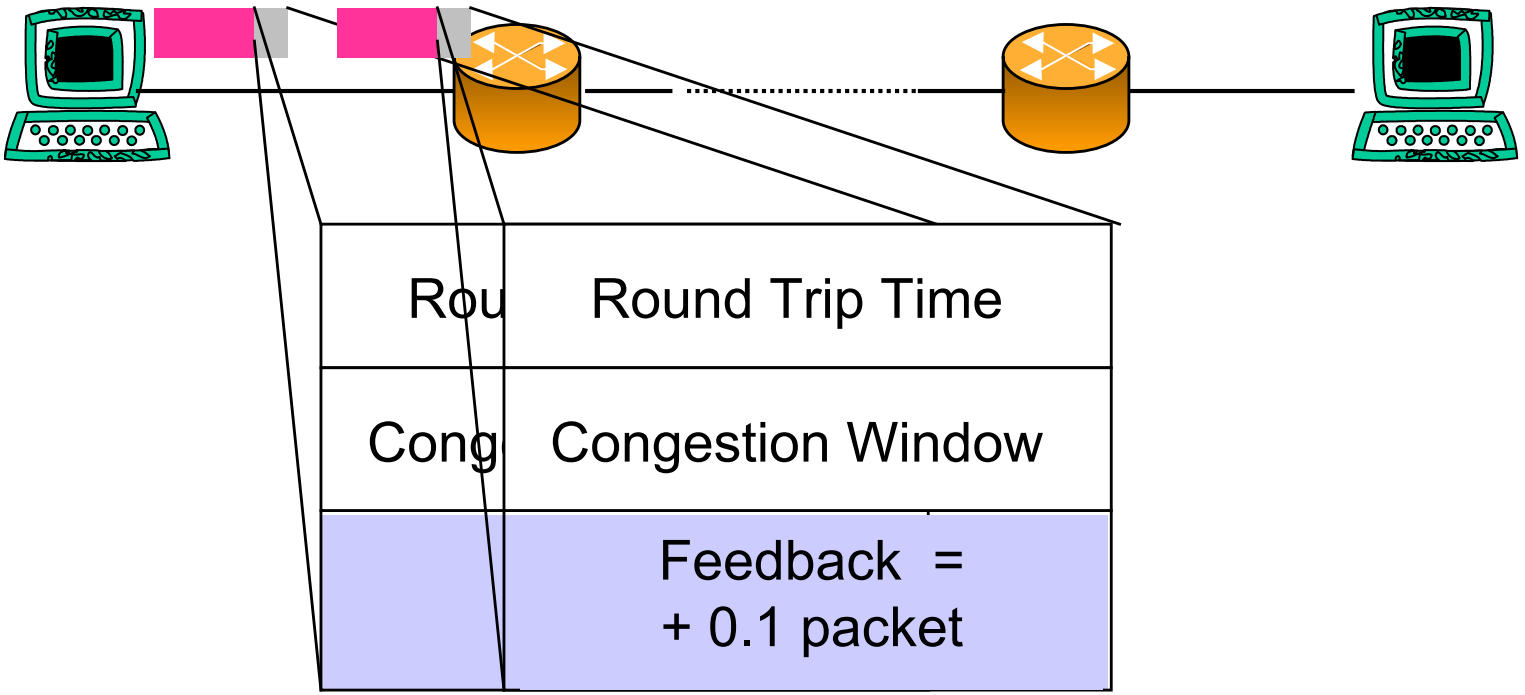
Zdroj: E. He, J. Leigh, O. Yu, T. A. DeFanti, "Reliable Blast UDP: Predictable High Performance Bulk Data Transfer," *IEEE Cluster Computing 2002*, Chicago, Illinois, Sept, 2002.

- A zahájení vysílání nastavenou rychlostí
- B konec vysílní
- C zaslání signálu DONE po řídícím kanálu, na nějž přijímající zareaguje zasláním masky dat, které dorazily
- D opětovné zaslání chybějících dat
- E-F-G dokončení přenosu

Přižemž body C a D se opakují tak dlouho, dokud nejsou doručena všechna data.

XCP

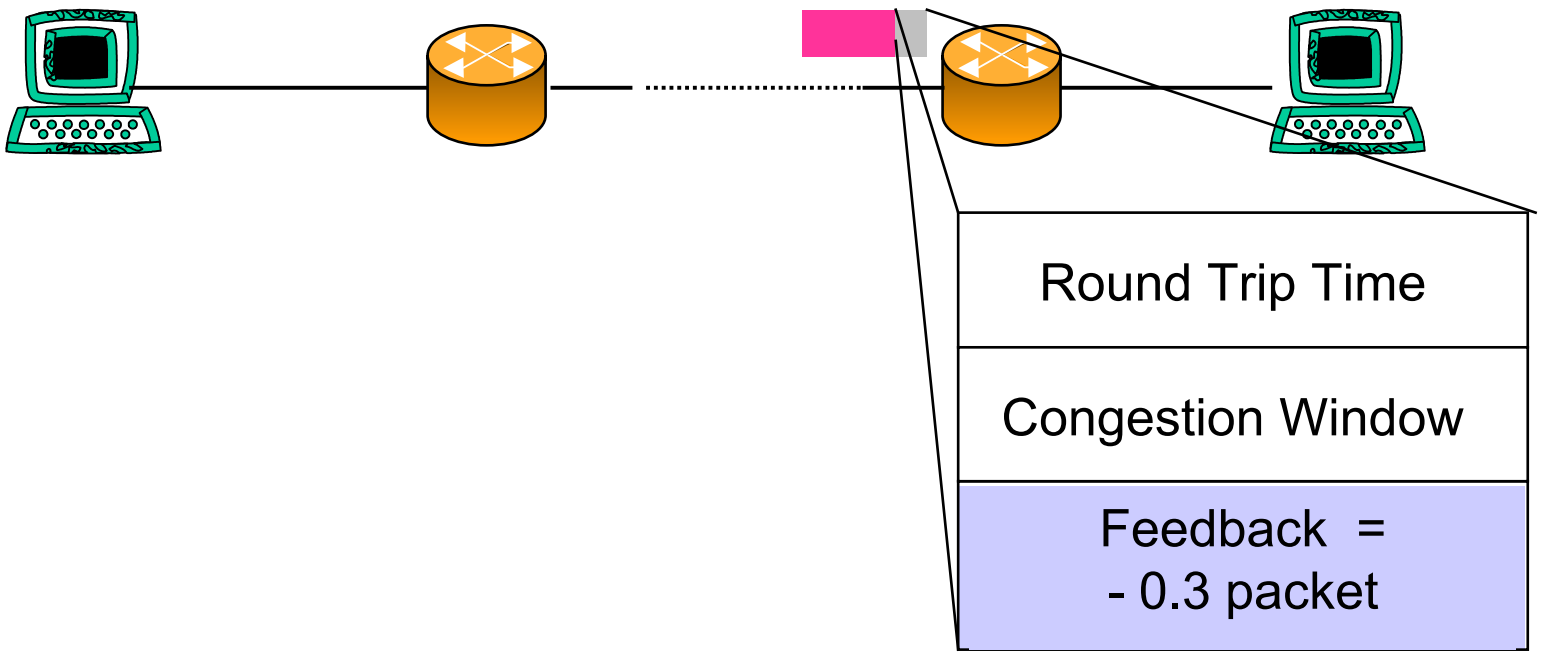
- zpětná vazba od směrovačů per paket



Congestion Header

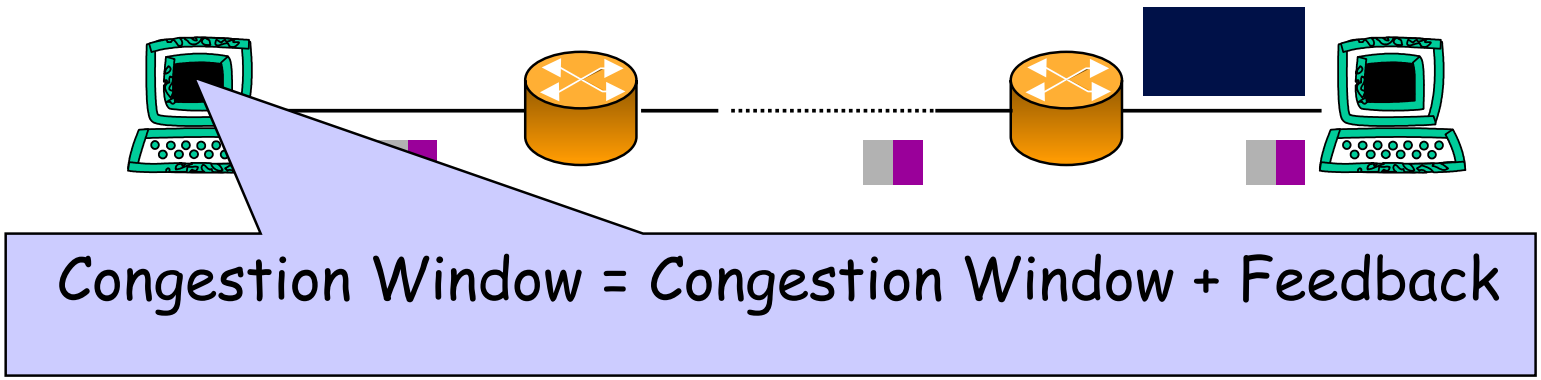
XCP

- zpětná vazba od směrovačů per paket



XCP

- zpětná vazba od směrovačů per paket



Jiné přístupy

- SCTP
  - víceproudový, multi-homed transport
  - `http://www.sctp.org/`
- DCCP
  - nezajištěný protokol (UDP) s řízením zahlcení kompatibilním s TCP
  - `http://www.ietf.org/html.charters/dccp-charter.html`
  - `http://www.icir.org/kohler/dcp/`

Jiné přístupy

- STP
  - založený na CTS/RTS
  - jednoduchý protokol pro snadnou implementaci v HW
  - bez sofistikovaného řízení zahlcení
  - `http://lwn.net/2001/features/OLS/pdf/pdf/stlinux.pdf`
- Reliable UDP
  - zajišťuje spolehlivé, in-order doručení (do maximálního počtu opakování retransmise)
  - RFC908 a RFC1151
  - původně vzniklo kvůli IP telefonii
  - konfigurace parametrů spojení per-spojení
  - `http://www.javvin.com/protocolRUDP.html`
- XTP (Xpress Transfer Protocol), …

Závěrečné poznámky

- Současný stav
  - víceproudové TCP se intenzivně používá (např. Gridové aplikace)
  - hledání cesty, jak bezpečně (zpětně kompatibilně) zajistit vývoj/nasazení post-TCP protokolů
  - používání agresivních protokolů na privátních/dedikovaných sítích a okruzích (např. λ-sítě CzechLight/CESNET2, SurfNet, CaNET*4, …)
  - implementace SCTP ve FreeBSD 7.0
  - implementace DCCP v Linuxu

Závěrečné poznámky

- Interakce s L3 (IP)
- Interakce se linkovou vrstvou
  - proměnné zpoždění a propustnost u bezdrátových sítí
  - optical burst switching
- Specifické per-flow stavy ve směrovačích
  - např. per-flow nastavení generovaných výpadků ($\longrightarrow$ E-TCP)
  - může pomoci krátkým tokům s vysokými přenosovými nároky (makro-bursty)
  - problém se škálovatelností a náklady :-(

Literatura


Jacobson V. “Congestion Avoidance and Control”, Proceedings of ACM SIGCOMM’88 (Standford, CA, Aug. 1988), pp. 314–329.
`ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z`


Allman M., Paxson V., Stevens W. “TCP Congestion Control”, RFC2581, Apr. 1999.
`http://www.rfc-editor.org/rfc/rfc2581.txt`


Brakmo L., Peterson L. “TCP Vegas: End to End Congestion Avoidance on a Global Internet”, IEEE Journal of Selected Areas in Communication, Vol. 13, No. 8, pp. 1465–1480, Oct. 1995. `ftp://ftp.cs.arizona.edu/xkernel/Papers/jsac.ps`


`http://www.web100.org`


Hacker T. J., Athey B. D., Sommerfield J. “Experiences Using Web100 for End-To-End Network Performance Tuning”
`http://www.web100.org/docs/ExperiencesUsingWeb100forHostTuning.pdf`

Literatura


Kelly T. "Scalable TCP: Improving Performance in Highspeed Wide Area Networks", PFLDnet 2003, http://datatag.web.cern.ch/datatag/pfldnet2003/papers/kelly.pdf, http://wwwlce.eng.cam.ac.uk/~ctk21/scalable/


Floyd S. "HighSpeed TCP for Large Congestion Windows", 2003, http://www.potaroo.net/ietf/all-ids/draft-floyd-tcp-highspeed-03.txt


BIC-TCP, http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/


Floyd S., Allman M., Jain A., Sarolahti P. "Quick-Start for TCP and IP", 2006, http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-quickstart-02.txt


Jin C., Wei D., Low S. H., Buhrmaster G., Bunn J., Choe D. H., Cottrell R. L. A., Doyle J. C., Newman H., Paganini F., Ravot S., Singh S. "FAST – Fast AQM Scalable TCP." http://netlab.caltech.edu/FAST/ http://netlab.caltech.edu/pub/papers/FAST-infocom2004.pdf


tsunami, http://www.anml.iu.edu/anmlresearch.html

Literatura


E. He, J. Leigh, O. Yu, T. A. DeFanti, "Reliable Blast UDP: Predictable High Performance Bulk Data Transfer," *IEEE Cluster Computing 2002*, Chicago, Illinois, Sept, 2002.


Stephen Hemminger, "TCP testing – Preventing global Internet warming", 2007, `http://www.linuxconf.eu/2007/papers/Hemminger.pdf`


David Hayes, Lawrence Stewart, Grenville Armitage, "Evaluating the FreeBSD 9.x Modular Congestion Control Framework's Performance Impact", *CAIA Technical Report 110228A*, February 2011, p. 5, `http://caia.swin.edu.au/reports/110228A/CAIA-TR-110228A.pdf`

Další studijní materiály

- Workshopy PFLDnet 2003–2006
  - `http://datatag.web.cern.ch/datatag/pfldnet2003/program.html`
  - `http://www-didc.lbl.gov/PFLDnet2004/`
  - `http://www.ens-lyon.fr/LIP/RESO/pfldnet2005/`
  - `http://www.hpcc.jp/pfldnet2006/`
  - `http://wil.cs.caltech.edu/pfldnet2007/`
- Strány s příspěvky prof. Sally Floyd
  - `http://www.icir.org/floyd/papers.html`
- RFC3426 – “General Architectural and Policy Considerations”

`http://www.hamilton.ie/net/eval/results_HI2005.pdf`