# Experimenty s QoS mechanizmom **WRED**

**Topológia:** štandardná..

PC1 (WS+I/O graphs,statistics, D-ITG+logger) - R1 --- R2 - PC2 (WS+I/O graphs,statistics, D-ITG+logger)

**D-ITG generovanie**:

* 4x TCP toky ~ 50 kbps
* 1x UDP tok ~ 50 kbps
* Každý jednotlivý tok, či TCP ci UDP generujeme s **oneskorením aspoň 5 sekúnd** po predošlom spustenom toku, t.j. prvý oneskoríme o 5s, druhý o 10s, atď. (zaznamenajte si potom aj v akom poradí ste ich spuštali - v kontexte AF značiek, ktoré prídu neskôr)
  + Chceme vidieť čo spôsobí každý ďalší tok s už existujúcimi tokmi sieťovej prevádzky
* Pokiaľ nebudú viditeľné straty, zvážte zvýšenie tokov na dvojnásobok alebo viac.
* Ideálne by bolo keby ste opäť spravili pre každý scenár (budú 4: FIFO, CBWFQ, CBWFQ+WRED, CBWFQ+WRED+ECN) aj dva varianty, z ktorých v jednom budú toky s menšou intenzitou, ale v súčte nad kapacitu linky 128kbps, a v druhom budú toky s vyššou intenzitou a v súčte tiež nad kapacitu linky, ale výrazne nad

**Aké parametre a štatistiky chceme sledovať:**

* **Wireshark**:
  + Sliding window
    - ako a či sa mení veľkosť TCP plávajúceho okna, či koncové stanice reagujú na stav zahltenia
      * odsledujeme pole window size v hlavičkách IP paketov
      * Je možnosť pozrieť aj: Statistics - TCP stream graphs – Windows Scalling
  + IO graphs – intenzita paketov v kbps – priebeh v čase
  + Statistics - zistiť či je tu niečo vhodné na sledovanie (zaznameajte do reportu) – resp. min. porovnať výsledky odtiaľto s tými, ktoré nám poskytne logger v D-ITG - skúška správnosti
  + Pre scenár s ECN - preskúmame IP a TCP hlavičku:
    - IP: ECN (00, **01**, **10**, **11**)
    - TCP: ECN-echo (0, **1**) a CWR (0, **1**)
      * Prehľadávanie množstva odchytených paketov môže byť na dlho, preto využite vhodné filtre... celkom určite sa dá vyhľadať špecifický paket s konkrétnu hodnotou ECN, a podobne aj špecifický TCP segment..
* **DITG**:
  + logger – oneskorenie (min, max, avg), straty
* **Na smerovači pre výstupné rozhranie**:
  + Show interface... (drops v output queue, typ queuing-u, a iné)
  + Show policy-map interface... (počet zahodených paketov v jednotlivých triedach, aj percentuálne vyjadrenie prípadne cez pravdepodobnosť, wred, queueing, ..)
* Chceme potvrdiť/vyvrátiť hypotézu:   
  ***"Pri použití WRED, by sme mali zaregistrovať menšie straty ako bez WRED."*** 
  + Pri FIFO zahodíme viac paketov, pretože sa vždy čaká až kým sa naplní celý front.
  + Pri WRED – v predstihu keď sa ešte len blíži naplnenie frontu – začína náhodne zahadzovať pakety – tak reagujú viaceré TCP privieraním okna – čo zmenšuje zahltenie a tým aj straty (zahodené pakety).
  + WRED + ECN – prvý smerovač, kde sa blíži k zahlteniu ešte nezahadzuje paket, ale ho preznačí **v IP hlavičke** (posledné 2 bity v ToS bajte, za 6-timi DSCP bitmi) z 01 alebo 10 na **11** – čím signalizuje cieľu, že na ceste bolo zahltenie a ten pošle zdroju **TCP** pokyn (flag **ECN-Echo** (ECE) – reakcia na oznámenie zahltenia) na zmenšenie okna, zdroj mu potvrdí zmenšenie TCP flagom (**Congestion Window Reduced** (CWR) – potvrdzujem zmenšenie okna). Pozn.: ak ide paket označený 11 cez druhý smerovač, tak ten ak má tiež zahltený svoj interface a vyjde mu, že by mal paket preznačiť, paket 11 už zahodí.

**Klasifikácia a značkovanie:**

Všetko vieme robiť zrkadlovo, ak stíhame, aby si každý z dvojice vyskúšal všetko, testy potom treba robiť jednotlivo, pre jedenaj druhý smer, t.j. raz som receiver a raz sender.

Každý na svojom smerovači nastaví na vstupe z LAN klasifikáciu paketov (zrejme stačí ACLkom rozlíšiť prevádzku podľa cieľového portu – toho, ktorý používame v DITG, v realite by to bola nejaké rozumnejšia klasifikácia) a ich značkovanie:

* 4 toky TCP: AF11, AF12, AF 31, AF 33
* 1 UDP tok: AF21

**Scenáre - experimenty:**

Pri každom scenári nasadím na výstupe zo smerovača (v smere k susednému smerovaču) istý typ frontovej stratégie (FIFO/CBWFQ) a istý typ QoS mechanizmu (nič/WRED/WRED+ECN), pri každom nasledonom scenári, treba predošlý zrušiť a nasadiť nový, prevažne riešenie je cez policy mapy:

1. FIFO
   1. Na vstupe síce značkujeme, ale v tomto scenári značky nevyužívame, a všetky toky budú na výstupe zo smerovača ukladané do jednej FIFO fronty
   2. Defaultne je zapnuté WFQ, treba ho vypnúť (no fair-queue)
   3. Treba si uvedomiť, že TCP vie reagovať na zahadzované pakety, zníži rýchlosť vysielania a veľkosť okna, a UDP nereaguje, t.j. stále posiela dáta rovnakou rýchlosťou, t.j. uchmatne si viac BW.
   4. Tu by sme mali byť schopní vidieť jav „globálna synchronizácia TCP tokov“: keď sa front naplní, smerovač začne zahadzovať pakety všetkých TCP tokov a všetky vysielacie stanice prudko zmenšia okná (receive window). Tým klesne aj súčtový tok prichádzajúci na rozhranie smerovača a front sa mierne vyprázdni, k stratám nedochádza a vysielače dostávajú potvrdenia na pakety ktoré vysielajú. Následne znova všetky stanice zväčšia okná, celkový tok postupne rastie, a opakuje sa stav zahltenia a zahadzovanie paketov.
      1. Avšak... v OS Windows je už implementovaná ochrana proti globálnej synchronizácii TCP tokov, preto nie je jednoduché ju zaznamenať, a pomerne ťažšie je demonštrovať výhodu WRED a WRED+ECN.
         1. Bonusová úloha: ako konkrétne je v OS Windows implementovaná ochrana proti synchronizácii TCP tokov?
2. CBWFQ
   1. **WFQ** = režim v ktorom sa pre každú novú konverzáciu vytvára samostatný front (a z tých frontov sa cyklicky vyberá, takže každý sa v danom kole dostane na rad – ku všetkým tokom férové – ponúka rovnaký bandwidth. My spravíme CBWFQ, my si budeme špecifikovať ktorá prevádzka bude v ktorej fronte:
   2. Na vstupe značkujeme, a tieto značky využijeme na rozlíšenie prevádzky do tried (na **výstupe**), pričom každá trieda bude mať svoju vlastnú FIFO frontu:
      1. Trieda1 – AF11
      2. Trieda2 – AF12
      3. Trieda3 – AF21
      4. Class-default – vsetko ostatné, čiže pre nás AF31 a AF33
         1. Každej triede (1,2,3 aj class-default) vyhradíme minimálnu šírku pásma 25% (bandwidt percent...)
3. CBWFQ+WRED
   1. Pre class-default spustíme mechanizmus WRED
4. CBWFQ+ECN-based WRED
   1. Na počítačoch vo Windowse treba povoliť/zapnúť ECN, aby Windows vytváral TCP spojenia s podporou ECN. Windows bude potom v IP hlavičke označovať ECN bity na 01 alebo 10. Protokol TCP bude potom vedieť reagovať na správy ECN prenášané v hlavičke IP paketu a ktoré mení pri zahltení smerovač.
      * 1. ECN je rozšírením k IP protokolu a TCP, definovaný v RFC 3168, umožňuje end-to-end notifikáciu zahltenia siete bez zahadzovania paketov. ECN je voliteľná funckia, ktorú možno použiť iba vtedy, keď ju podporujú obe kocové stanice a chcú ju využiť, a tiež keď zariadenia v sieti ju podporujú (smerovače).
        2. Windows od verzie Windows Server 2008 a Windows Vista podporujú ECN pre TCP, ale je defaultne vypnuté, zapnúť to treba v príkazovom riadku príkazom:   
           ***netsh int tcp set global ecncapability=enabled***  
           Vypnúť to možno podobne: ....  ***=disabled***
        3. Odsledujte, či vidíte toto značkovanie vo WS. Čokoľvek vygenerujem, odíde to s ECN bitmy v IP hlavičke 01 alebo 10. Uveďte do reportu ako to bolo vo vašom prípade, či 01 či 10.
           1. Bonusová úloha: zistite, od čoho závisí, či sa použije 01 alebo 10 – je to vec operačného systému?
   2. Na smerovači na výstupnom rozhraní pre class-default spustíme mechanizmus WRED+ECN
      1. Pri náhodnom zahadzovaní paketov WRED mechanizmom, ich smerovač teraz označí dvomi bitmi ECN (Explicit Congestion Notification) v IP hlavičke (hneď za 6 bitmy pre DSCP) na 11 (a nezahodí ako pri predošlom scenári), čím signalizuje prijímaču, že je na ceste zahltenie. Prijímač následne požiada vysielač, aby zmenšil okno, využije na to flag v TCP hlavičke ECE (ECN-Echo – t.j. výzva pre zdroj dát, aby zmenšil svoje okno, t.j. používa sa v smere Destination → Source). Vysielač mu v ďalšej TCP hlavičke nastvením bitu CWR (Congestion Window Reduced) potvrdí, že okno zmenšil, t.j. používa sa v smere Source → Destination.
      2. Poznámka: dnes už ECN nemá veľký význam, pretože OS už majú implementovanú istú ochranu proti synchronizácií TCP tokov. WRED ako taký význam má.