Damian Nowacki

Konfiguracja i badanie wybranych mechanizmów zapewnienia parametrów jakościowych w systemach sieciowych

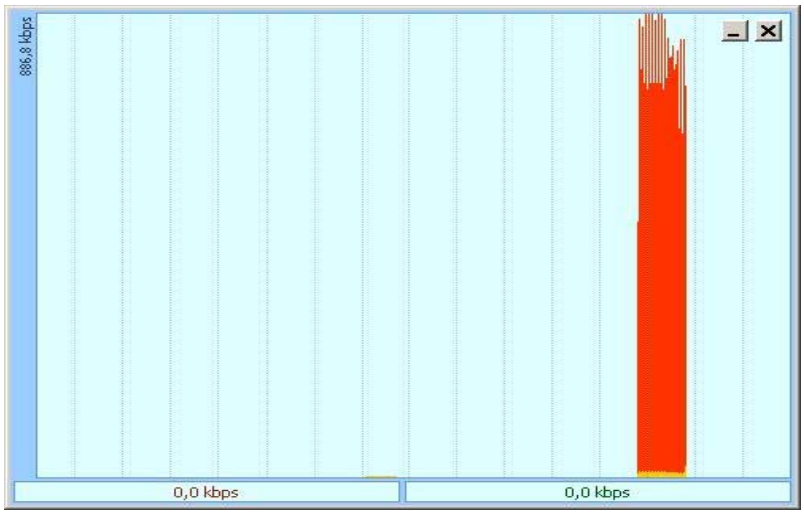
Fragmenty pracy dyplomowej inżynierskiej

Do użytku tylko jako materiał do ćwiczeń laboratoryjnych na Wydziale Elektrycznym Politechniki Białostockiej

Promotor pracy: dr inż. Andrzej Zankiewicz

4.2.1 Ograniczenie przepustowości metodą Shaping dla ruchu z odpowiednim polem DSCP

W kolejnym badaniu sprawdzono poprawność działania ograniczenia przepustowości dzięki przypisaniu odpowiedniego pola DSCP. Utworzono trzy klasy high, average oraz low. Każda z klas różniła się przypisanym do niej polem DSCP. W policy_map utworzono zasadę do której przypisano te trzy klasy. Dzięki kształtowaniu *shaping average* przypisano różne prędkości dla odpowiednich klas. Przepustowość łącza między dwoma routerami była ustawiona na poziomie 1 Mb/s



Rys. 4.8 Pełna przepustowość połączenia między routerami R1 i R2 przed testem klas

Tab. 4.1 Zestawienie trzech utworzony klas z przypisanymi różnymi polami dscp oraz z różnymi ograniczeniami przepustowości

Nazwa klasy	Przypisane pole DSCP	Ograniczona przepustowość
High	Af43	500 kb/s
Average	Af22	200 kb/s
Low	Af11	100 kb/s

Poniżej zostanie zaprezentowana konfiguracja routera brzegowego R1. Należało w tym kroku utworzyć trzy nowe klasy. Do każdej z klas przypisać kryterium dopasowania po polu dscp. Utworzone klasy należało połączyć z utworzoną zasadą *policy-map Shape*. W policy-map dzięki mechanizmie *shape average* ograniczyć przepustowość do odpowiednich wartości. Na końcu na interfejsie szeregowym przyłączyć utworzoną zasadę dla ruchu wchodzącego.

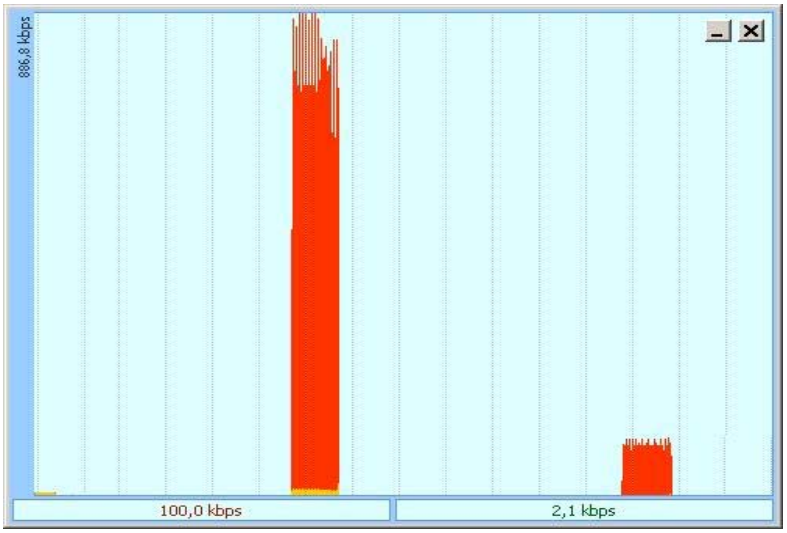
```
R1(config)# class-map high
R1(config-cmap)# match dscp af43
R1(config)# class-map average
R1(config-cmap)# match dscp af22
R1(config)# class-map low_klasa
R1(config-cmap)# match dscp af11
R1(config)#policy-map Shape
R1(config-pmap)#class high_klasa
R1(config-pmap-c)#shape average 500000
R1(config-pmap)#class average
R1(config-pmap-c)#shape average 200000
R1(config-pmap)#class low_klasa
R1(config-pmap-c)#shape average 100000
R1(config)#interface serial 0/0
R1(config-if)#service-policy output Shape
```

Na rysunku 4.9 przedstawiono konfigurację routera R1. Ukazano utworzenie trzech klas high, average oraz low. Przypisano odpowiednią przepustowość dla danych klas metodą *shape average.*

```
Class-map: high (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match:  dscp af43 (38)
  Match: access-group 101
  Traffic Shaping
       Target/Average   Byte   Sustain   Excess    Interval  Increment
         Rate           Limit  bits/int  bits/int  (ms)      (bytes)
       500000/500000    3000   12000     12000     24        1500

    Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
    Active Depth                         Delayed   Delayed   Active
    -      0         0         0         0         0         no

Class-map: average (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match:  dscp af22 (20)
  Match: access-group 101
  Traffic Shaping
       Target/Average   Byte   Sustain   Excess    Interval  Increment
         Rate           Limit  bits/int  bits/int  (ms)      (bytes)
       200000/200000    2000   8000      8000      40        1000

    Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
    Active Depth                         Delayed   Delayed   Active
    -      0         0         0         0         0         no

Class-map: low (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match:  dscp af11 (10)
  Match: access-group 101
  Traffic Shaping
       Target/Average   Byte   Sustain   Excess    Interval  Increment
         Rate           Limit  bits/int  bits/int  (ms)      (bytes)
       100000/100000    2000   8000      8000      80        1000

    Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
    Active Depth                         Delayed   Delayed   Active
    -      0         0         0         0         0         no

Class-map: class-default (match-any)
  1 packets, 24 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: any
```

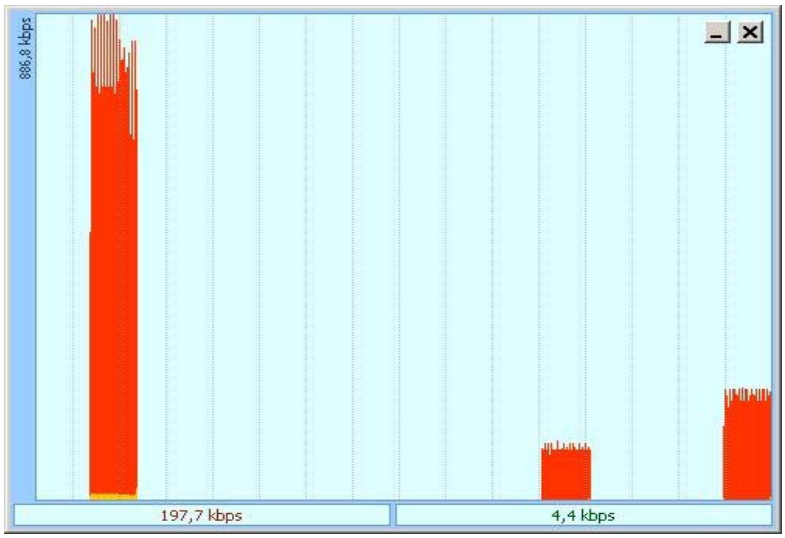
Rys. 4.9 Zrzut ekranu komputera po dokonaniu konfiguracji routera R1

Po odpowiednim skonfigurowaniu routera R1, należało uruchomić program iperf3. W wierszu poleceń po stronie serwera wpisano komendę: *iperf3 -4 -s,* w tym momencie uruchomiono nasłuchiwanie serwera. Po stronie klienta wpisano trzy różne komendy dla trzech różnych parametrów. Pierwszą z nich była komenda: *iperf3 -4 –c 192.168.2.2 –t 30 –S 40*. Wartość przy zmiennej –S 40 to wartość ToS dziesiętnie odpowiadająca polu DSCP af11. Parametr –t określał czas trwania ruchu. Po uruchomieniu komendy za pomocą programu DU Meter zaobserwowano ograniczenie do 100 kb/s.



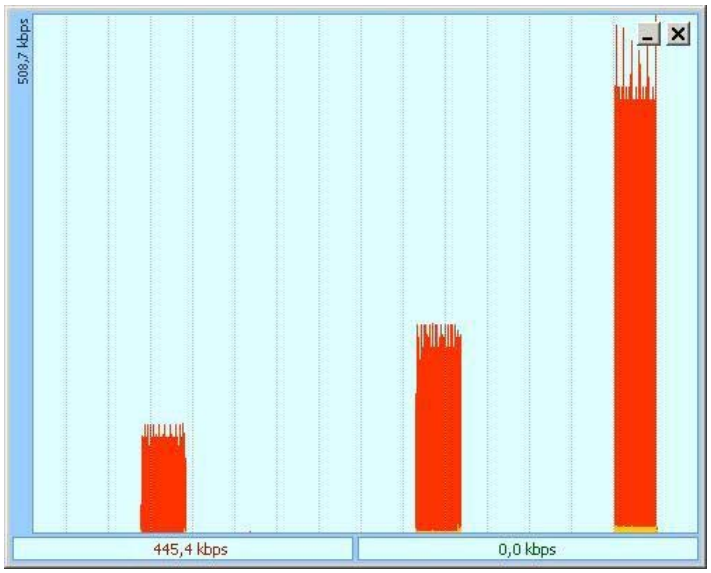
Rys. 4.10 Ograniczenie przepustowości dla klasy low do wartości 100 kb/s

Druga komenda dla klasy average wyglądała następująco: *iperf3 -4 –c 192.168.2.2 –t 30 –S 80.* Wartość przy parametr –S 80 to wartość ToS dziesiętnie odpowiadająca polu DSCP: af22. Po uruchomieniu komendy za pomocą programu DU Meter zaobserwowano ograniczenie do 200 kb/s



Rys. 4.11 Ograniczenie przepustowości dla klasy average na poziomie 200 kb/s

Trzecia komenda dla klasy high wyglądała następująco: *iperf3 -4 –c 192.168.2.2 –t 30 –S 152*. Wartość przy parametrze –S 152 to wartość ToS dziesiętnie odpowiadająca polu DSCP af43. Po uruchomieniu komendy za pomocą programu DU Meter zaobserwowano ograniczenie do 500 kb/s.



Rys. 4.12 Ograniczenie przepustowości dla klasy high na wartość 445 kb/s

4.2.2 Podsumowanie i wnioski

Przypisanie odpowiedniego pola DSCP dla ruchu, gwarantowało zdeklarowaną prędkość transmisji. Metoda *Shaping average* zapewniła określoną wartość średniej szybkości łącza. Algorytm *Shaping* rozpoznając w ruchu wyjściowym odpowiedne pole DSCP, poprzez włączenie mechanizmu kształtowania ograniczał średnią prędkość łącza. Istotne w tym zadaniu było to, by sprawdzić czy komputery rozpoznają ruchu generowany przez program iperf3. Dodano wpis w rejestrze o nazwie *DisableUserTOSSetting* i typie REG_DWORD o wartości zero na obu komputerach, co skutkowało rozróżnianiem pola DSCP przez obje stacje. Pominięcie tego kroku spowodowałoby, nierozróżnianiem przez program iperf wartości pola DSCP i przypisaniu mu parametru równego 0. Ograniczenie dla klasy najwyższej high osiągnęło szybkość 445 kb/s, nie osiągając zakładanej prędkości 500 kb/s w czasie wykonania zrzutu ekranu. Pakiety jednakże osiągały w tej klasie także prędkość pobierania do 508 kb/s, różnice polegały na chwilowych obciążeniach w łączu.

4.5.1 Zbadanie mechanizmu kolejkowania Custom Queuing.

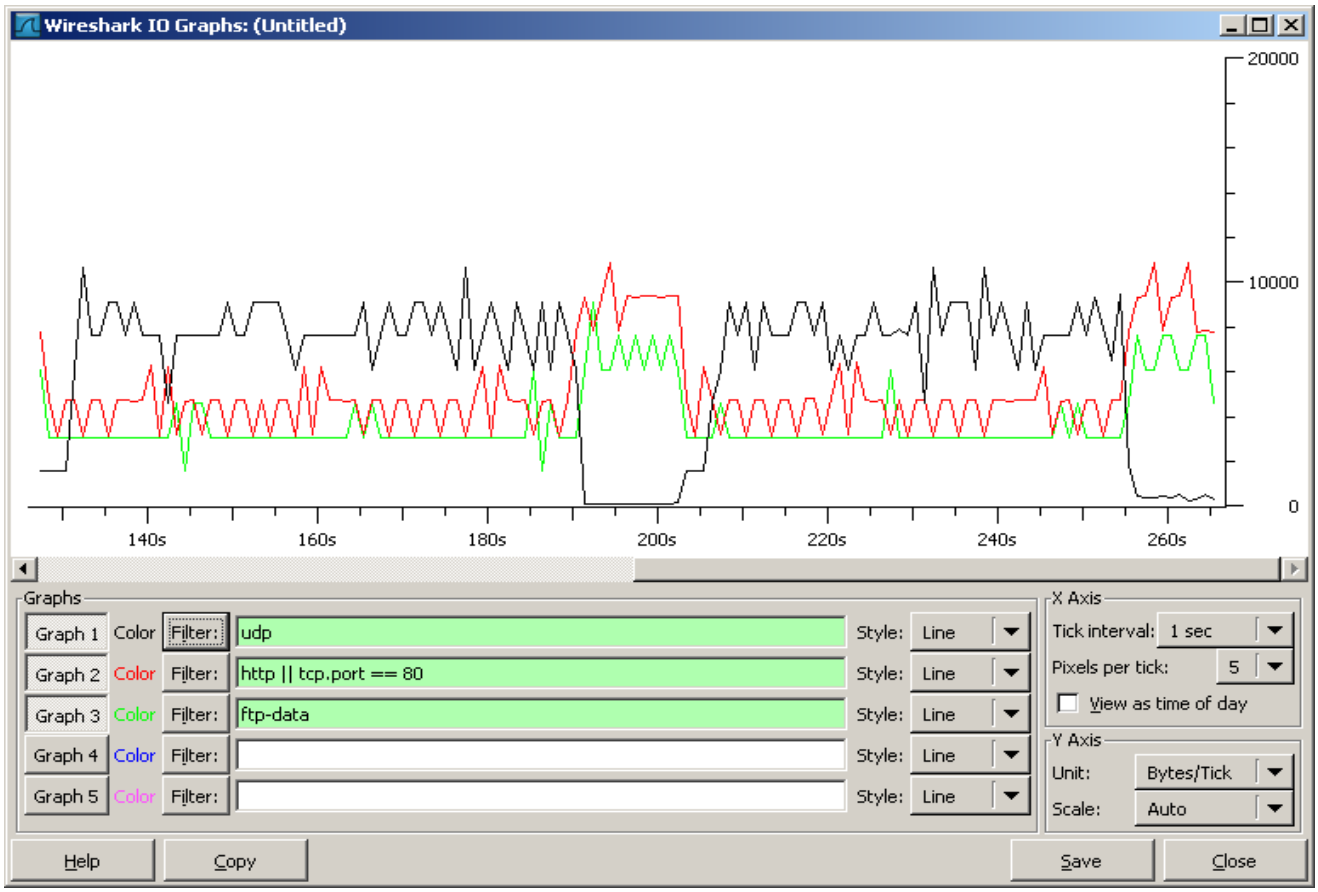
Wykonanie testów mechanizmu kolejkowania Custom Queuing, polegał na ograniczeniu przepustowości dla trzech różnych transmisji. W tym wypadku użyto trzech sposobów emulowania ruchu. Ruch protokołu udp, przesłano za pomocą programu iperf3 z komputera klienta na serwer. Dzięki programowi Total Commander i połączenia serwera ftp, ściągano plik testowy *test_ftp.zip* z komputera klienta na komputer serwera. Trzecim ruchem był ruch protokołu http. Wykorzystano tutaj plik testowy *test_http.zip*, który znajdował się na komputerze klienta. Przy wpisaniu w przeglądarce następującego adresu *http://192.168.1.2/test_http.zip* pobierano plik testowy http przez port 80. Poniżej zostanie zaprezentowana konfiguracja routera R1.

```
access-list 101 permit udp any any
access-list 102 permit tcp any eq www any
access-list 103 permit tcp any eq ftp-data any
queue-list 1 protocol ip 3 list 101
queue-list 1 queue 3 byte-count 8000
queue-list 1 protocol ip 4 list 102
queue-list 1 queue 4 byte-count 4000
queue-list 1 protocol ip 5 list 103
queue-list 1 queue 5 byte-count 3000
queue-list 1 default 6
queue-list 1 queue 6 byte-count 1000
interface serial 0/0
custom-queue-list 1
end
```

Na początku należało utworzyć trzy różne listy dostępowe. Pierwszą z list o numerze 101, była lista umożlwiająca ruch protokołu udp od każdego hosta do każdego. Lista 102 umożliwiała ruch protokołu http, także dla każdego hosta. Ostatnia z list umożliwiała ruch ftp-data w tej samej konfiguracji . Następnie utworzono pierwszą kolejkę dla listy 101, która ograniczała ruch dla protokołu udp do poziomu 8000 bajtów. Druga kolejka ogranicza ruch dla protokołu http do 4000 bajtów. Trzecia z utworzonych kolejek ogranicza ruch dla ftp-data do 3000 bajtów. Następnie utworzono ostatnią kolejkę domyślną, która będzie ograniczała niedopasowany ruch do 1000 bajtów. Należy tutaj podkreślić, że ograniczono łącze pomiędzy routerami R1 i R2 do poziomu 128 kb/s

Poniżej zostanie zaprezentowana komenda sprawdzająca poprawność dodania mechanizmu CQ oraz list dostępowych do routera brzegowego R1.

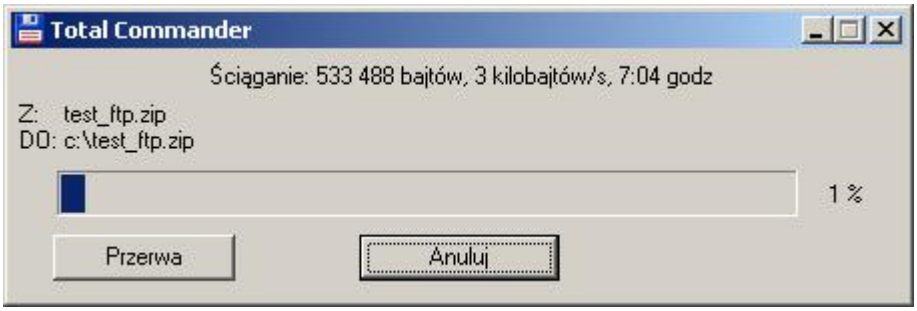
```
R1#sh queueing custom
Current custom queue configuration:
List   Queue  Args
1      6      default
1      3      protocol ip          list 101
1      4      protocol ip          list 102
1      5      protocol ip          list 103
1      3      byte-count 8000
1      4      byte-count 4000
1      5      byte-count 3000
1      6      byte-count 1000
R1#sh interfaces serial 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Internet address is 172.16.1.1/24
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 4/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  CRC checking enabled
  Last input 00:00:00, output 00:00:02, output hang never
  Last clearing of „show interface” counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 9472
  Queueing strategy: custom-list 1
```



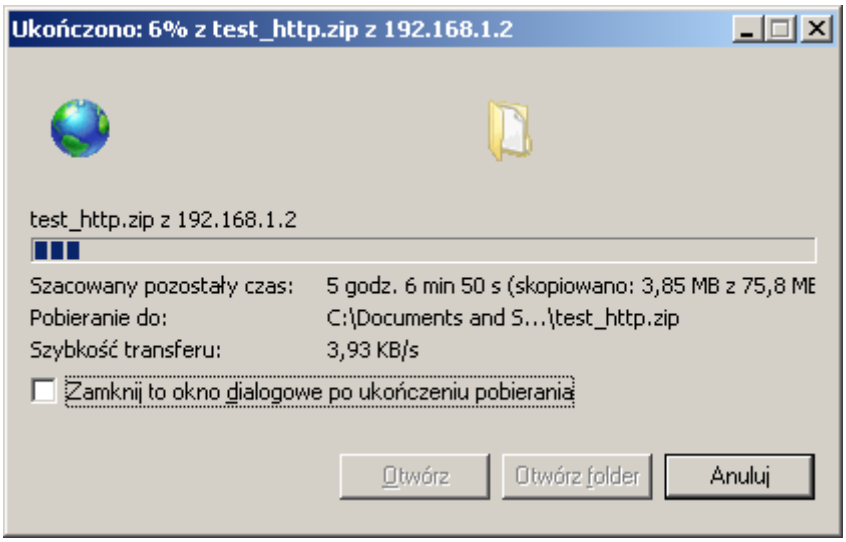
Rys. 4.17 Wynik kolejkowania CQ trzech różnych typów ruchu. Ruch protokołu udp reprezentowany jest kolorem czarnym. Ruch protokołu http kolorem czerwonym, ruch protokołu ftp kolorem zielonym. Na osi poziomej jest określony czas transmisji na osi pionowej wartość pobierania pakietów w bajtach

Na rysunku 4.17 zaprezentowano ograniczenie ruchu do określonych prędkości. Ruch protokołu udp reprezentowany linią o kolorze czarnym w 220s transmisji osiąga około 8 kB/s. Ruch protokołu http reprezentowany przez linię koloru czerwonego, w tej samym czasie transmisji osiąga poziom około 4 kB/s. Zielona linia reprezentująca ruch ftp-data osiąga prędkość rzędu 3 kB/s. Na podstawie wykresu można potwierdzić poprawność działania mechanizmu CQ.

Na poniższych zrzutach zostaną zaprezentowane dokładne prędkości chwilowe pobierania pliku testowego http oraz pliku ftp-data.



Rys. 4.18 Prędkość pobierania pliku test_ftp.zip przez komputer serwera



Rys. 4.19 Prędkość pobierania pliku test_http.zip przez komputer serwera

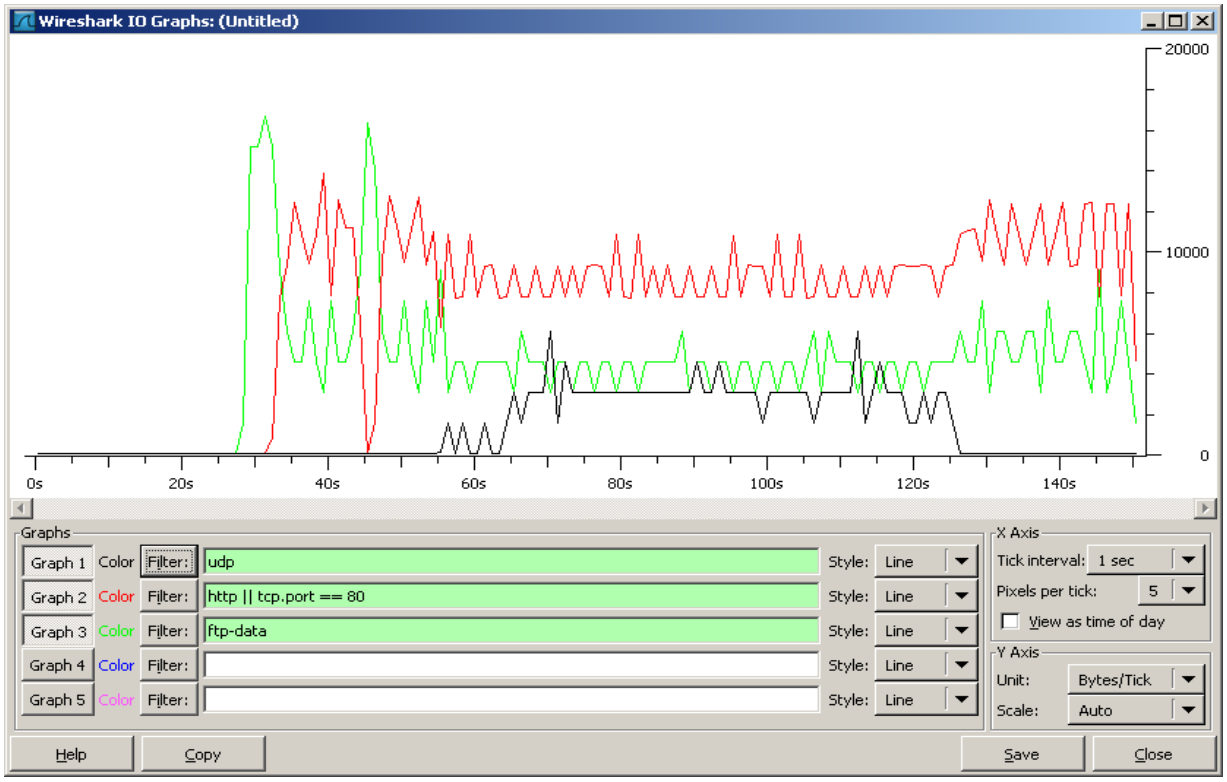
Następny test polegał na zamianie ograniczenia prędkości różnego rodzaju ruchu. W tym teście ruch protokołu http powinien osiągać przepustowość 8 kB/s. Ruch ftp-data powinien mieć prędkość transmisji na poziomie 4 kB/s, a ruch protokołu udp tylko 3 kB/s. Bez zmian zostaje ruch domyślny przypisujący niedopasowany ruch do tego warunku.

```
access-list 101 permit udp any any
access-list 102 permit tcp any eq www any
access-list 103 permit tcp any eq ftp-data any
queue-list 1 protocol ip 3 list 102
queue-list 1 queue 3 byte-count 8000
queue-list 1 protocol ip 4 list 103
queue-list 1 queue 4 byte-count 4000
queue-list 1 protocol ip 5 list 101
queue-list 1 queue 5 byte-count 3000
queue-list 1 default 6
queue-list 1 queue 6 byte-count 1000
interface serial 0/0
custom-queue-list 1
end
```

Poniżej zostanie zaprezentowana komenda sprawdzająca poprawność dodania mechanizmu CQ oraz list dostępowych do routera brzegowego R1.

```
R1#sh queueing custom
Current custom queue configuration:
List    Queue   Args
1       6       default
1       3       protocol ip          list 102
```

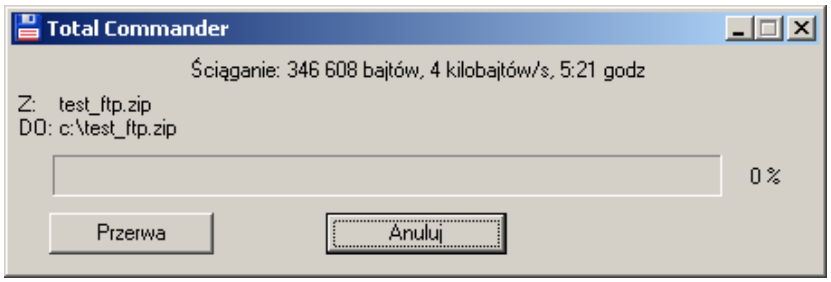
```
1       4       protocol ip          list 103
1       5       protocol ip          list 101
1       3       byte-count 8000
1       4       byte-count 4000
1       5       byte-count 3000
1       6       byte-count 1000
```



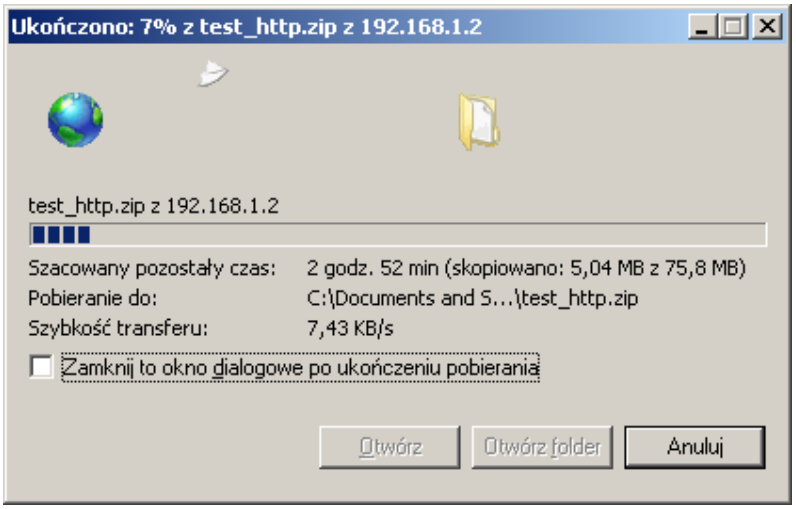
Rys. 4.20 Wynik kolejkowania CQ trzech różnych transmisji. Ruch protokołu udp reprezentowany jest kolorem czarnym. Ruch protokołu http kolorem czerwonym, ruch protokołu ftp kolorem zielonym. Na osi poziomej jest określony czas transmisji na osi pionowej wartość pobierania pakietów w bajtach

Na rysunku 4.20 przedstawiono ograniczenie ruchu do określonych prędkości. Ruch protokołu udp reprezentowany linią o kolorze czarnym, w 80s transmisji tym razem osiąga wartość około 3 kB/s. Ruch protokołu http reprezentowany przez linię koloru czerwonego, w tym samym czasie transmisji osiąga poziom najwyższy około 8 kB/s. Zielona linia reprezentująca ruch ftp-data, osiąga przepustowość na poziomie 4 kB/s. Na podstawie wykresu, można i w tym przypadku potwierdzić poprawność działania mechanizmu CQ.

Na poniższych rysunkach zostaną zaprezentowane dokładne prędkości chwilowe pobierania pliku http oraz pliku ftp-data



Rys. 4.21 Prędkość pobierania pliku test_ftp.zip przez komputer serwera



Rys. 4.22 Prędkość pobierania pliku test_http.zip przez komputer serwera

4.5.2 Podsumowanie i wnioski

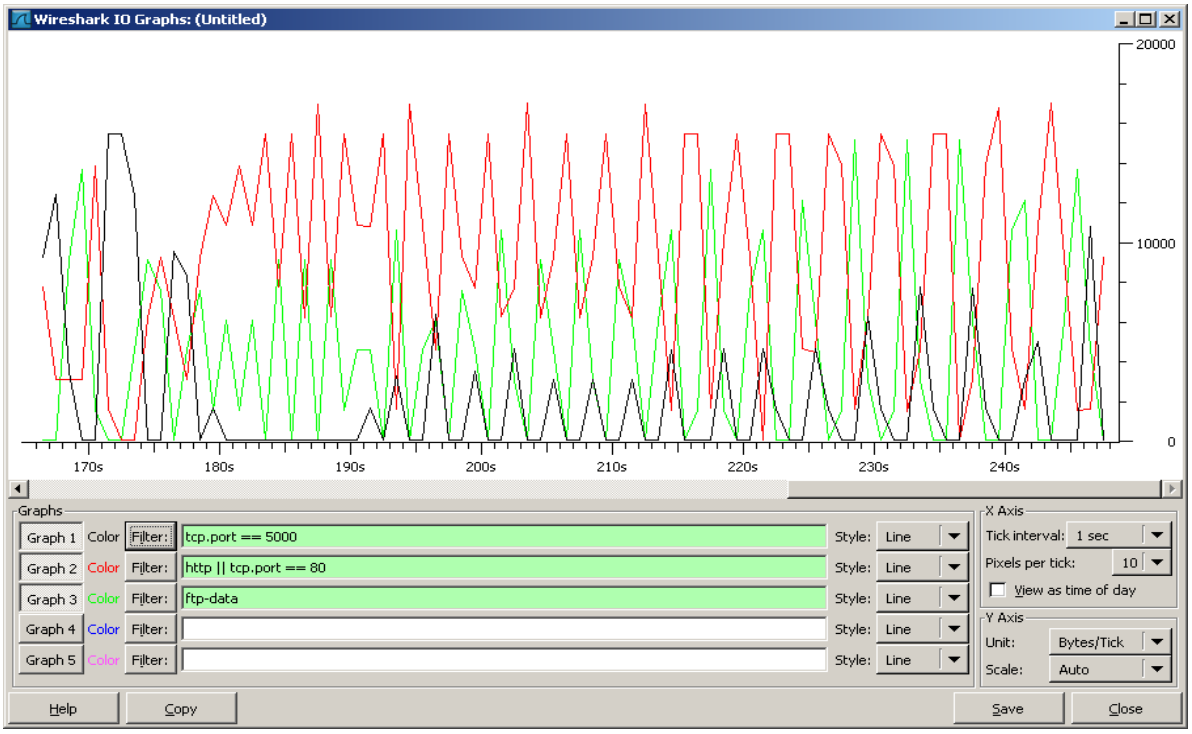
Mechanizm Custom Queuing umożliwia sterowanie pasmem na interfejsie routera. Na podstawie wyników widać, że kolejkowanie gwarantuje każdemu typowi ruchu przydzieloną mu przepustowość, nawet w przypadku obciążenia. Mechanizm kolejkowania CQ automatycznie uruchamia szesnaście kolejek dla ruchu generowanego przez aplikacje oraz jedną dla potrzeb systemowych. Należy pamiętać o zdefiniowaniu kolejki domyślnej własnoręcznie. Zwiększanie liczby kolejek powoduje wzrost obciążenia routera, który musi dokonać rozdziału strumienia.

4.7.1 Zbadanie mechanizmu kolejkowania Weighted Fair Queuing.

W kolejnym pomiarze sprawdzono działanie mechanizmu kolejkowania WFQ. Na routerach, które są dostępne w laboratorium o danym systemie ios, nie ma wielu opcji jeśli chodzi o mechanizm WFQ. Jedyną opcją jaka jest możliwa do włączenia to polecenie *fair-queue*, która powoduje włączenie mechanizmu. Badanie będzie polegało na wysyłaniu trzech transmisji między komputerami z włączonym i wyłączonym mechanizmem wfq. Tak jak wcześniej ruch pobierania pliku ftp-data oraz protokołu http. Trzecim rodzajem będzie ruch protokołu tcp z przypisanym krytycznym polem dscp o wartości ToS 184, generowanym poprzez program iperf3. W pierwszej kolejności zostanie wyłączony mechanizm wfq, ponieważ mechanizm ten jest domyślnie włączony na routerach o łączach 2 Mbps lub interfejsach o niższej przepustowości. Następnie mechanizm ten zostanie włączony i obie konfiguracje zostaną zaobserwowane w programie Wireshark IO Graphs.

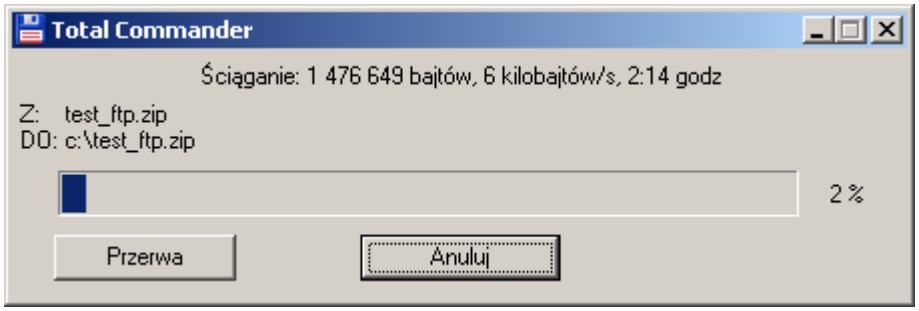
Na początku w ustawieniach interfejsu szeregowego wyłączono sprawiedliwe kolejkowanie. Przepustowość łącza w tym badaniu wynosiła 128 kb/s.

```
R1(config)#interface serial 0/0
R1(config-if)#no  fair-queue
```

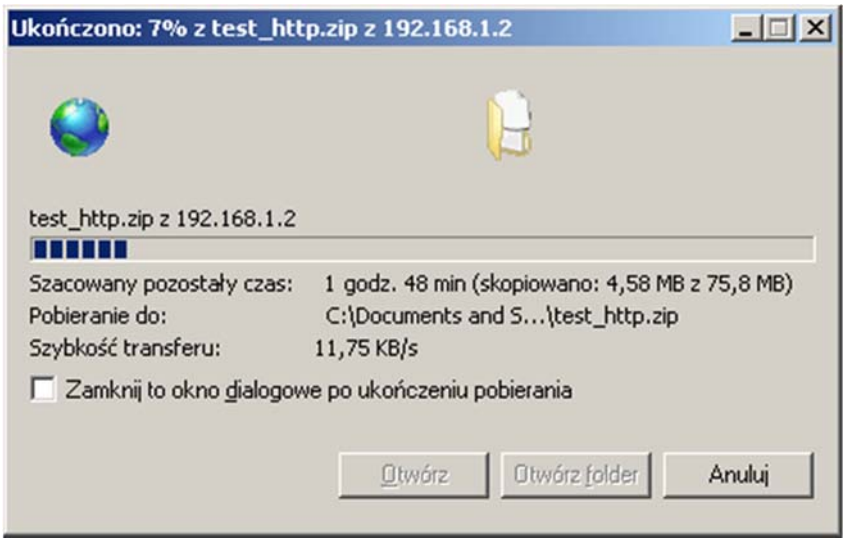


Rys. 4.24 Zrzut ekranu komputera z programu IO Graphs z wyłączonym mechanizmem wfq. Ruch protokołu tcp po porice 5000 reprezentowany jest kolorem czarnym. Ruch protokołu http kolorem czerwonym, ruch protokołu ftp kolorem zielonym

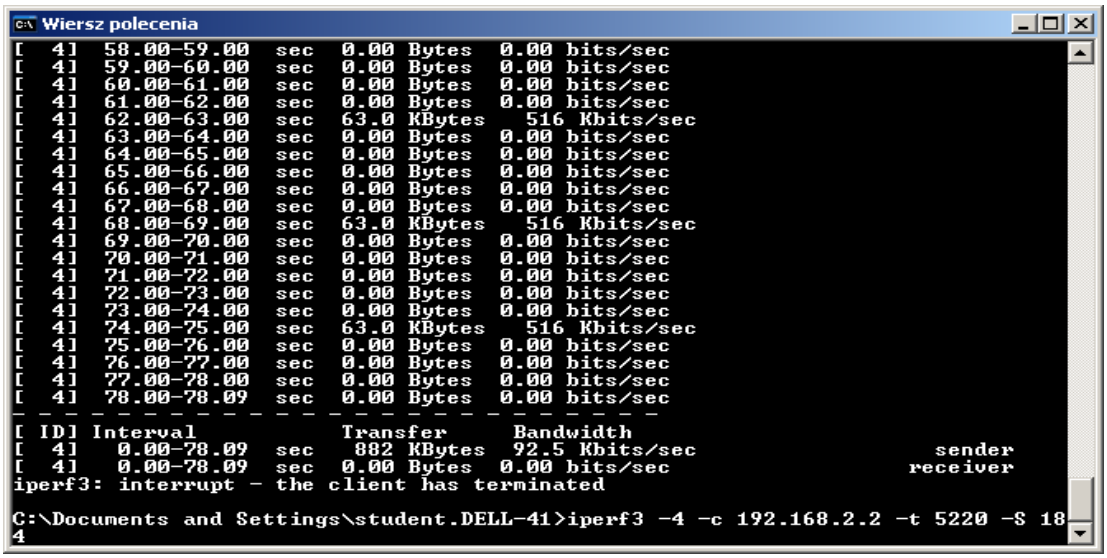
Na zrzucie ekranu rysunku 4.24 zaprezentowano trzy różnego rodzaju ruchu sieciowe. Czarna linia reprezentuje ruch protokołu tcp wysyłany przez program iperf3 z przypisanym polem dscp. Port 5000 jest to port na którym nasłuchuje serwer, ustawiony w programie iperf3 po stronie serwera. Następnym ruchem sieciowym jest ruch protokołu http po porcie 80, reprezentowany poprzez czerwoną linię. Na końcu ruch ftp-data, którą reprezentuje linia zielona. Jak widać w pierwszym teście ruch jest niesprawiedliwie traktowany. Pobieranie pliku *test_http.zip* osiąga najwyższą przepustowość z tych trzech rodzajów ruchu. Przesyłany ruch protokołu tcp z programu ipef3 osiąga najmniejszą przepustowość średnio 32 kb/s.



Rys. 4.25 Pobieranie pliku pliku ftp-data http przez komputer serwera o prędkości 6k/s przy wyłączonym mechanizmie wfq



Rys. 4.26 Pobieranie pliku http przez komputer serwera o prędkości 11,75 kb/s przy wyłączonym mechanizmie wfq

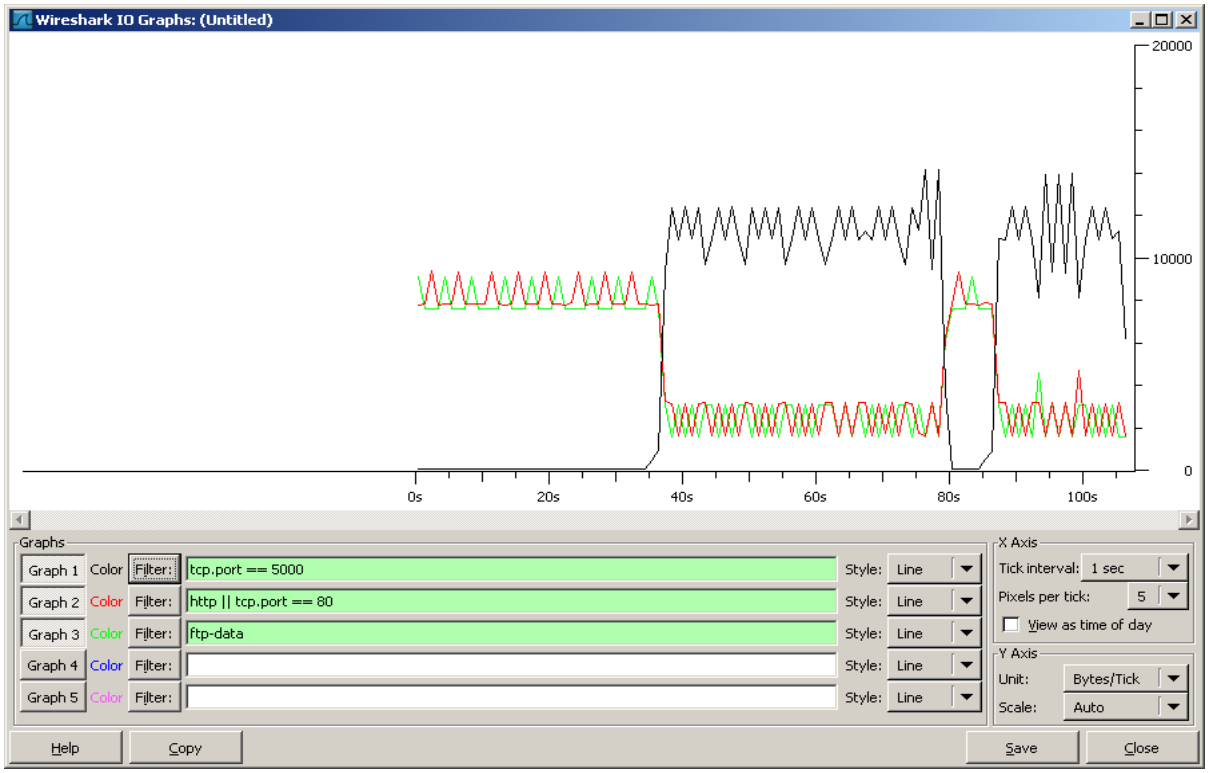


```
Wiersz polecenia
[  4]  58.00-59.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  59.00-60.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  60.00-61.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  61.00-62.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  62.00-63.00  sec  63.0 KBytes   516 Kbits/sec
[  4]  63.00-64.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  64.00-65.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  65.00-66.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  66.00-67.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  67.00-68.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  68.00-69.00  sec  63.0 KBytes   516 Kbits/sec
[  4]  69.00-70.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  70.00-71.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  71.00-72.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  72.00-73.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  73.00-74.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  74.00-75.00  sec  63.0 KBytes   516 Kbits/sec
[  4]  75.00-76.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  76.00-77.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  77.00-78.00  sec  0.00 Bytes  0.00 bits/sec
[  4]  78.00-78.09  sec  0.00 Bytes  0.00 bits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-78.09  sec   882 KBytes  92.5 Kbits/sec                  sender
[  4]   0.00-78.09  sec  0.00 Bytes  0.00 bits/sec                  receiver
iperf3: interrupt - the client has terminated

C:\Documents and Settings\student.DELL-41>iperf3 -4 -c 192.168.2.2 -t 5220 -S 18
4
```

Rys. 4.27 Zrzut ekranu komputera z programu iperf3 po stronie klienta z wysyłaniem ruchu tcp w stronę serwera o przypisanym polu ToS 184. Transmisja odbywa się do portu 5220

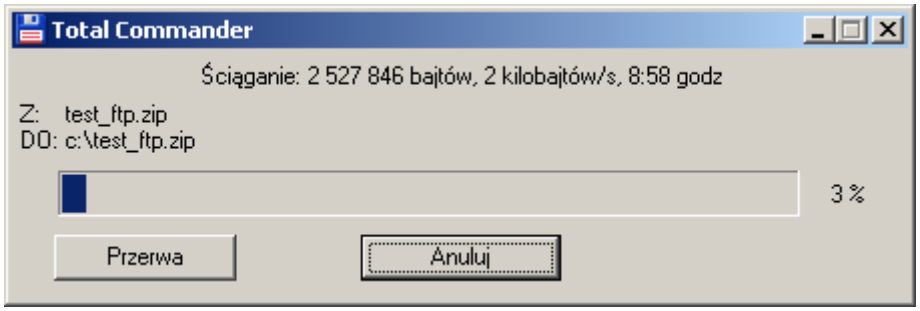
W następnym kroku badania skonfigurowano router brzegowy R1. Na interfejsie szeregowym 0/0 włączono mechanizm wfq. Od tego momentu router powinien tak dostosowywać przepustowość, by ruch o najwyższej wartości pola dscp był priorytetem nad innym o mniejszej wartości pola dscp.

```
R1(config)#interface serial 0/0
R1(config-if)# fair-queue
```

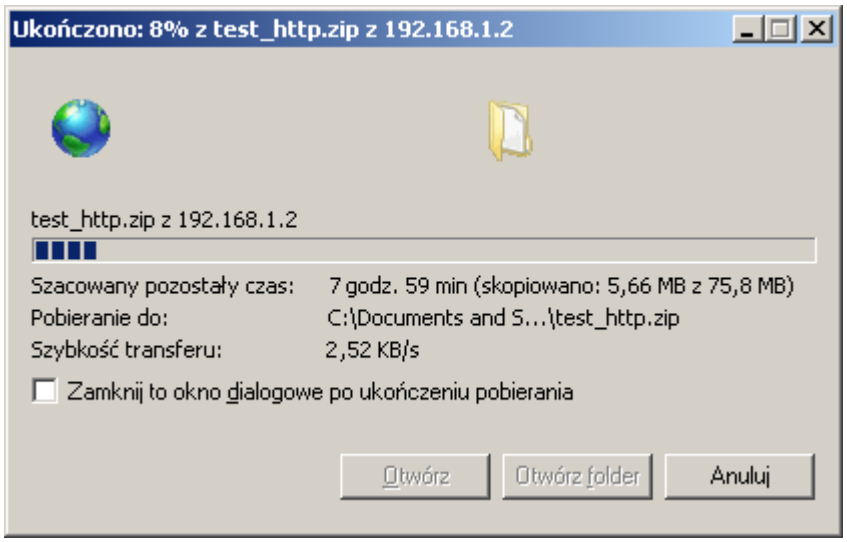


Rys. 4.28 Zrzut ekranu programu IO Graphs z włączonym mechanizmem wfq. Ruch protokołu tcp po porcie 5000 reprezentowany jest kolorem czarnym. Ruch protokołu http kolorem czerwonym, ruch protokołu ftp kolorem zielonym

Na zrzucie ekranu rysunku 4.28 zaprezentowano trzy różne rodzaje ruchu sieciowego. Czarna linia reprezentuje ruch protokołu tcp wysyłany przez program iperf3 z przypisanym polem dscp. W tym teście widać działanie mechanizmu wfq. Od 40s, gdy włączono przesyłanie ruchu protokołu tcp z najwyższym polem dscp ,ruch ten posiada najwyższą przepustowość. Ruch ten osiąga prędkość na poziomie 96 kb/s. Dwa pozostałe ruchy o niższym priorytecie osiągają prędkość o wiele niższą niż w pierwszym teście. Wyniki prędkości obu transmisji przedstawiono poniżej.



Rys. 4.29 Prędkość pobierania pliku test_ftp.zip przy włączonym mechanizmie wfq



Rys. 4.30 Prędkość pobierania pliku test_http.zip przy włączonym mechanizmie wfq

4.7.2 Podsumowanie i wnioski

Powyższy pomiar służył temu, by przy przekazywanie pakietów router uwzględniał wartości pola ToS i DSCP. Jeśli weighted fair queuing nie jest włączone, mechanizmem domyślnym kolejkowania jest fifo, które jest mało efektywne. Mechanizm wfq przydziela ruchowi o wyższym priorytecie większą cześć pasma, jednocześnie uniemożliwia mu przejęcia pełnej przepustowości niezależnie od wartości pola ToS.

4.3.1 Zbadanie mechanizmu Token Bucket Filter na ograniczonej przepustowości za pomocą mechanizmu Shaping.

Do przebadania tego mechanizmu potrzebne jest utworzenie klasy oraz zasady. Utworzone zostaną dwie klasy jedna o nazwie *average_klasa* druga o nazwie *peak_klasa.* Nazwy te odpowiadają ograniczeniu przepustowości dzięki mechanizmie *shape*. Przy odpowiednim dostosowaniu dwóch parametrów wiadra z żetonami Bc i Be można zaobserwować działanie tegoż algorytmu. Parametr Bc odpowiada ile żetonów zostanie pobrane z kubełka w danym okresie czasu, co powoduje wysłanie danej liczby bitów na wyjście Na przykład, gdy parametr ten jest ustawiony na poziomie 2000 bitów. Liczba 2000 żetonów jest pobierana z kubełka co sekundę. Zadaniem tego pomiaru było zaobserwowanie jak dany parametr wpływa na kształtowanie ruchu. Przed wykonanie badań należy przedstawić co oznaczają poszczególne parametry CIR, Bc oraz Be:

• Committed Information Rate (CIR) określa, ile danych może być przesłane lub przekazane w średniej jednostce czasu.

• Committed Burst (Bc) wielkość ta określona w bitach lub bajtach na wielkość serii. Oznacza jak duży ruch może być wysłany w danej jednostce czasu.

• Excess Burst (Be) wielkość określona w bitach lub bajtach. Oznacza jak duży ruch może być wysłany, przekraczając limit określony przez wartość CIR.

W pierwszej części zadania badana będzie funkcja *shape average.* Skonfigurowanie tej funkcji spowoduje, że interfejs wysyła nie więcej danych niż rozmiar określony w parametrze Bc w danym interwale czasowym. Zarazem osiągając średnią szybkość nie wyższą niż CIR.

Poniżej zostanie zaprezentowana konfiguracja routera R1 dla przebadania pierwszego z mechanizmu *shape average*.

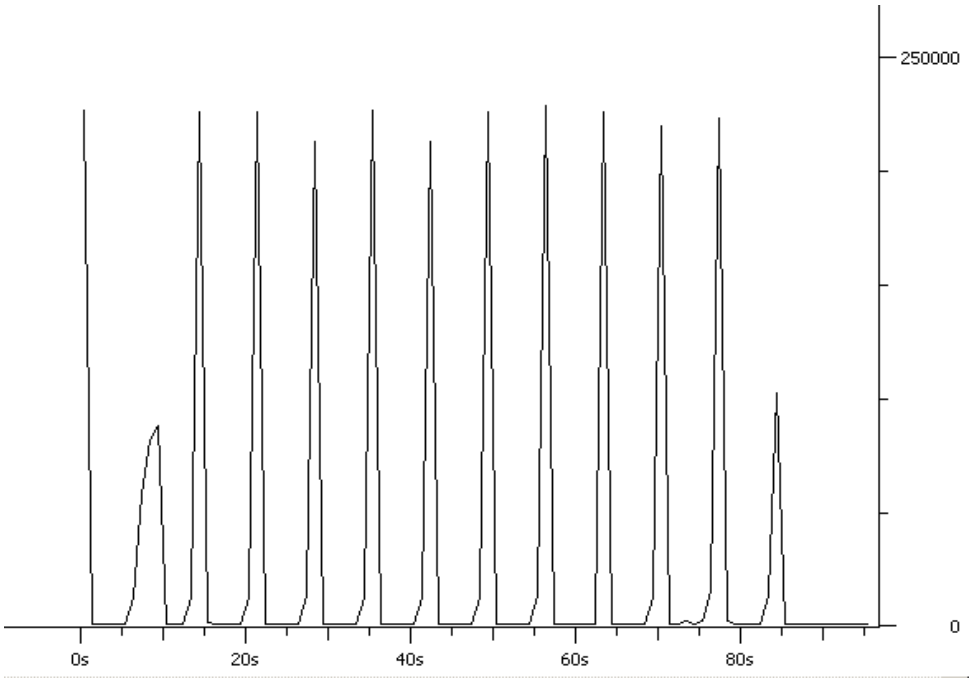
```
R1(config)#class-map average_klasa
R1(config-cmap)#match any
R1(config)#policy-map shape_zasada
R1(config-pmap)#class average_klasa
R1(config-pmap-c)#shape average 32000 224000 0
R1(config)#interface serial 0/0
R1(config-if)#service-policy output shape_zasada
```

Poniżej zostanie zaprezentowana nowo utworzona zasada *policy-map shape_zasada* dla mechanizmu average

```
R1#sh policy-map interface serial 0/0
```

```
Serial0/0
 Service-policy output: shape_zasada
   Class-map: average_klasa (match-all)
     0 packets, 0 bytes
     5 minute offered rate 0 bps, drop rate 0 bps
     Match: any
     Traffic Shaping
         Target/Average   Byte   Sustain   Excess    Interval  Increment
            Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           32000/32000     28000  224000    0         7000      28000
       Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
       Active Depth                         Delayed   Delayed   Active
       -      0         0         0         0         0         no
   Class-map: class-default (match-any)
     0 packets, 0 bytes
     5 minute offered rate 0 bps, drop rate 0 bps
     Match: any
```

W powyższym listingu widnieje ograniczenie przepustowości do 32 kb/s. Parametr Bc jest ustawiony na poziom 224 kb/s. Jest siedmiokrotnie większy od ograniczonej przepustowości. Dla tego parametr Tc (Interval Time) jest równy 7000 ms, co wynika ze wzoru $\mathrm{Tc} = \frac{Bc}{CIR}$. Tak ustawiony TBF będzie działał następująco. Kubełek będzie wypełniany 224000 żetonami co 7s, a więc ruch będzie mógł być wysyłany o wartości 224000 kb/s co 7 sekundy. By zbadać tą konfigurację użyto pliku testowego *test_ftp.zip*. Stronę klienta połączono za pomocą programu Total Commander ze stroną serwer. Serwer połączył się poprzez serwer ftp z komputerem klienta. Następnie należało pobrać plik testowy ze stacji klienta na serwer poprzez protokół ftp. Wynik pobierania wraz z włączonym mechanizmem TBF zaprezentowano na rysunku 4.13



Rys. 4.13 Wynik działania algorytmu po stronie serwera. Na osi poziomej jest określony czas transmisji na osi pionowej liczba wysłanych bitów

Na zrzucie ekranu 4.13 programu IO Graphs wireshark ukazano działanie algorytmu. Na początku, gdy kubełek jest wypełniony żetonami następuje przesłanie pełnej puli żetonów czyli 224000. Następnie kubełek był wypełniany kolejną pulą żetonów. W około 7 sekundzie, nastąpiło kolejne przesłanie pełnej puli żetonów. Ten proces był powtarzany dokładnie z interwałem czasowym równym 7 sekund.

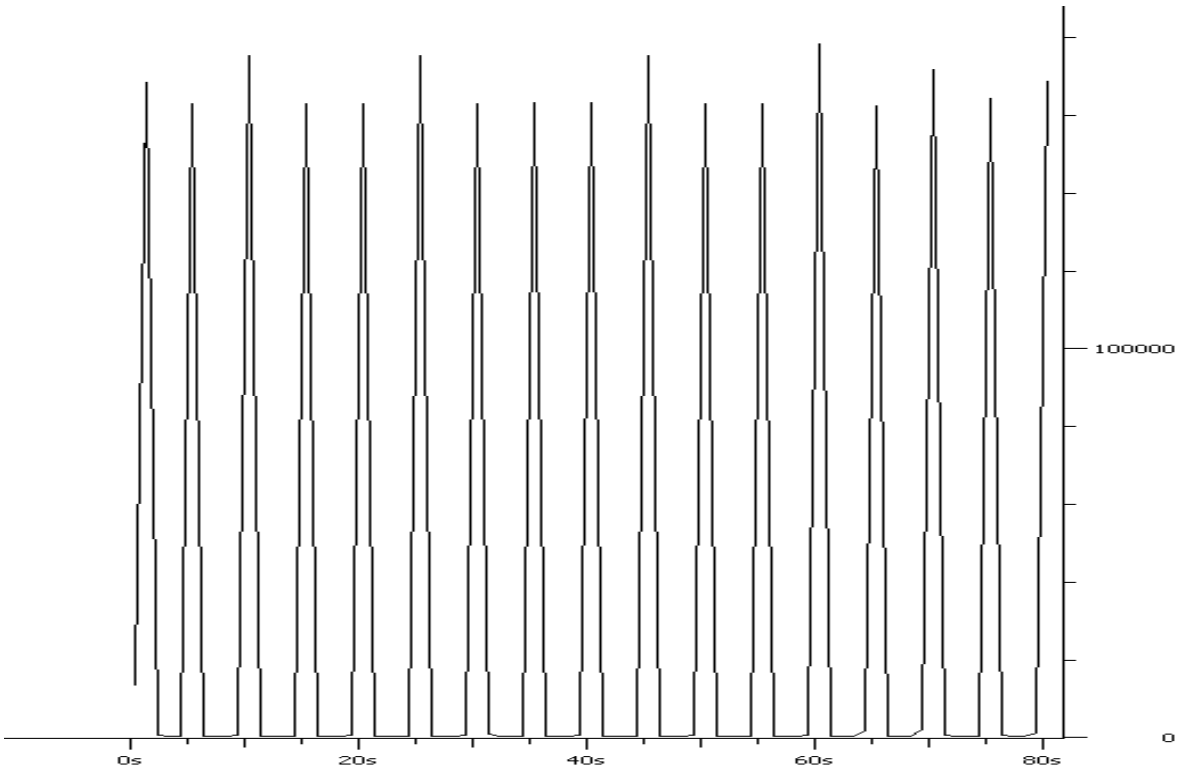
W drugiej części zadania badana będzie funkcja *shape peak.* Skonfigurowanie tej funkcji spowoduje, że interfejs będzie wysyłał wartość równą sumie dwóch parametrów Bc oraz Be w każdym przedziale czasu.

W kolejnym kroku zbadano mechanizm kształtowania s*hape peak*. Poniżej zostanie zaprezentowana konfiguracja routera R1 dla przebadania drugiego z mechanizmu.

```
R1(config)#class-map peak_klasa
R1(config-cmap)#match any
R1(config)#policy-map shape-zasada
R1(config-pmap)#class peak
R1(config-pmap-c)#shape peak 32000 160000 0
R1(config)#interface serial 0/0
R1(config-if)#service-policy output shape_zasada
```

Poniżej zostanie zaprezentowana nowo utworzona zasada *policy-map shape_zasada* dla mechanizmu peak

```
R1#show policy-map interface serial 0/0
 Serial0/0
  Service-policy output: shape_zasada
    Class-map: peak (match-all)
      28430 packets, 38868179 bytes
      5 minute offered rate 84000 bps, drop rate 73000 bps
      Match: any
      Traffic Shaping
           Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
            32000/32000     20000  160000    0         5000      20000
        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      0         3424      4266307   2948      3813421   no
    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```



Rys. 4.14 Wynik działania algorytmu peak po stronie serwera. Na osi poziomej jest określony czas transmisji na osi pionowej liczba wysyłanych bitów

Na powyższym zrzucie ekranu programu IO Graphs wireshark, ukazano działanie algorytmu *shape peak*. Na początku, gdy kubełek jest wypełniony żetonami następuje przesłanie pełnej puli żetonów czyli około 160000. Następnie kubełek był wypełniany kolejną pulą żetonów. W około 5 sekundzie nastąpiło, kolejne przesłanie pełnej puli żetonów. Ten proces był powtarzany dokładnie z interwałem czasowym równym 5 sekund.

Różnicą pomiędzy algorytmem *shape average* a *shape peak* jest taka iż, w pierwszym przypadku algorytm w odpowiednim czasie wyśle tylko określoną ilość żetonów nie przekraczając tej wartości. W *shape peak* sytuacja ta wygląda inaczej. Parametr PIR (Peak information rate) to szczytowa szybkość informacji, która jest określona jako przepływność lub szybkość łącza. Wartość maksymalna to suma parametrów Bc oraz Be, a wzór wygląda następująco $\mathrm{PIR} = \mathrm{Bc} + \mathrm{Be}$. W wypadku równości oby parametrów wzór $\mathrm{PIR} = \mathrm{CIR} * (1 + \frac{Bc}{Be})$ przyjmuje następującą formę

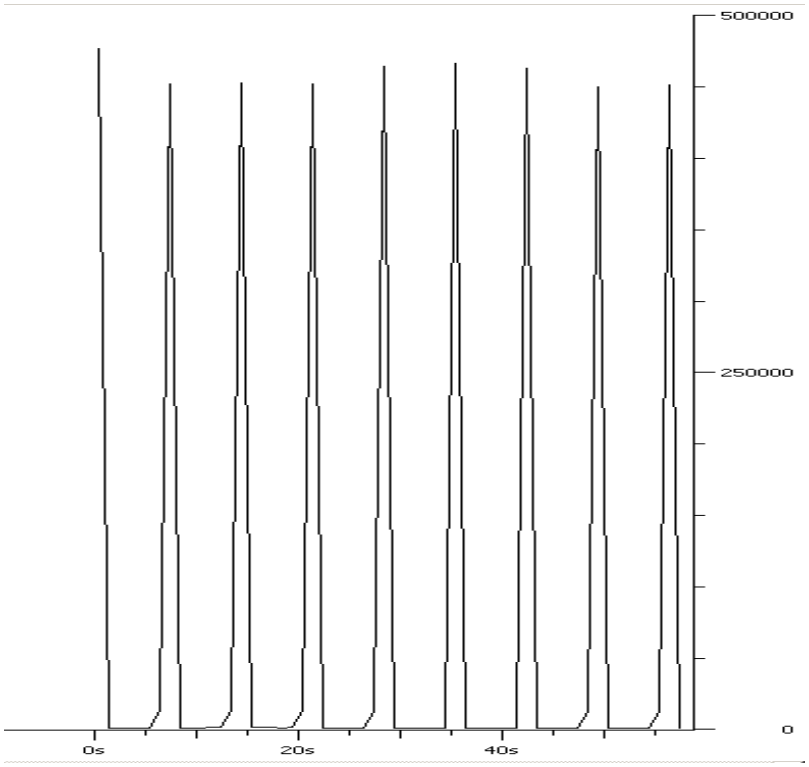
$\mathrm{PIR} = 2 * \mathrm{CIR}$. Taką sytuację zaprezentowano w kolejnym punkcie badań.

```
R1(config)#class-map peak_klasa
R1(config-cmap)#match any
R1(config)#policy-map shape_zasada
R1(config-pmap)#class peak_klasa
R1(config-pmap-c)#shape peak 32000 224000 224000
R1(config)#interface serial 0/0
R1(config-if)#service-policy output shape_zasada
```

Poniżej zostanie zaprezentowana nowo utworzona zasada *policy-map shape_zasada* dla mechanizmu *peak*. W tym teście parametry Bc oraz Be ustawiono na tym samym poziomie 160 kb/s.

```
R1#show policy-map interface serial 0/0
 Serial0/0
  Service-policy output: shape_zasada
    Class-map: peak (match-all)
      28430 packets, 38868179 bytes
      5 minute offered rate 84000 bps, drop rate 73000 bps
      Match: any
      Traffic Shaping
           Traffic Shaping
           Traffic Shaping
           Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
```

```
              Rate            Limit  bits/int  bits/int  (ms)      (bytes)
             64000/32000      56000  224000    224000    7000      56000
Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                           Delayed   Delayed   Active
        -      0         3424      4266307   2948      3813421   no
    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
                                   Match: any
```



Rys. 4.15 Wynik działania algorytmu peak z równymi parametrami Bc oraz Be po stronie serwera. Na osi poziomej jest określony czas transmisji na osi pionowej liczba wysłanych bitów

Na powyższym zrzucie ekranu programu IO Graphs wireshark, ukazano działanie algorytmu *shape peak*. Na początku, gdy kubełek jest wypełnione żetonami następuje przesłanie pełnej puli żetonów czyli około 448000. Następnie kubełek, było wypełniany kolejną pulą żetonów. W około 7 sekundzie nastąpiło kolejne przesłanie pełnej puli żetonów. Ten proces był powtarzany dokładnie z interwałem czasowym równym 7 sekund. Jak ukazano na rysunku 4.15 prędkość przesyłania w każdym interwale czasowym była równa około 450 kb/s. W tym wypadku algorytm *shape peak* pozwalał na transmisję o większej prędkości niż określony parametr CIR. Powodem tego był parametr Be, który pozwolił na zwiększenie pojemności kubełka z żetonami. Powiększenie to zarazem powoduje, możliwość przesłania większej ilości żetonów w każdym czasie Tc.

4.3.2 Podsumowanie i wnioski

Po wykonaniu zadania można potwierdzić działanie algorytmu TBF. Algorytm ten spełnia swoje zadania w różnych konfiguracjach. Wykonując te zadanie można zauważyć, istotną różnicę między kształtowaniem peak a kształtowaniem average. Pierwszy mechanizm kształtowania *peak* nie jest gwarancją przesłania pełnej puli żetonów. Jest jednak w stanie przesłać dwukrotnie więcej, dzieje się to tylko wtedy, gdy łącze nie jest obciążone. W powyższym teście w łączu odbywała się tylko jedna transmisja, więc możliwe było działanie *shape peak* bez obciążenia. Drugi mechanizm kształtowania *Shape average,* jest kształtowaniem z gwarancją, zapewnia średnią prędkość transmisji bez utraty pakietów.