Sieci i chmury teleinformatyczne

Laboratorium nr 1 - Płaszczyzna danych sieci

Rafał Dadura, Juliusz Kuzyka

Październik 2023

Spis treści

1 Cel laboratorium

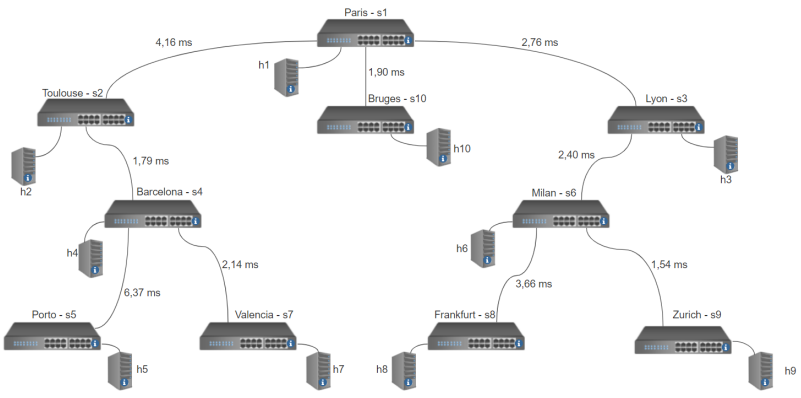
Głównym celem tego laboratorium jest zdobycie doświadczenia w emulacji sieci przy wykorzystaniu narzędzia Mininet oraz eksploracja sposobów analizy cezów ruchu w sieci za pomocą narzędzia do generowania ruchu o nazwie iperf.

2 Zadanie 1

Na początku przygotowaliśmy plan infrastruktury łączącej 10 miast europejskich, gdzie każde z tych miast posiada po jednym hoście i switchu. Większe miasta mają więcej łączy od miast mniejszych. Połączenia pomiędzy switchami zostały zobrazowane na mapie wraz z odległością, między jaką się znajdują.

Rysunek 1: Plan infrastruktury

W celu odzwierciedlenia realistycznych opóźnień w naszej sieci, wynikających z szybkości propagacji sygnału optycznego w światłowodzie wynoszącej około 200 000 km/s, dokonaliśmy przeliczenia odległości między miastami. Przeskalowaliśmy te odległości, uwzględniając charakter połączeń światłowodowych, poprzez pomnożenie ich przez $\sqrt{2}$, a następnie podzielenie przez 200 000 m/ms (np. Paris - Lyon $\frac{391km\sqrt{2}}{200\frac{km}{ms}} = 2,76ms$).



Rysunek 2: Topologia sieci z pokazanymi hostami i switchami, korzystając z https://www.smartdraw.com/

Następnie stworzyliśmy odpowiedni skrypt w Pythonie, który odzwierciedla stworzoną przez nas infrastrukturę. Najważniejsze funkcje, które zawierały się w naszym kodzie to:

```
Paris = self.addHost("h1")
Toulouse = self.addHost("h2")
...

s1 = self.addSwitch("s1")
s2 = self.addSwitch("s2")
...

self.addLink(Paris, s1, delay='0.1ms', bw=100)
self.addLink(Toulouse, s2, delay='0.1ms', bw=100)
...

self.addLink(s1, s2, delay='4.16ms', bw=100, loss=3.4, max_queue_size=2000)
self.addLink(s1, s3, delay='2.76ms', bw=100, loss=2.9, max_queue_size=2000)
self.addLink(s1, s10, delay='1.9ms', bw=100, loss=5.1, max_queue_size=2000)
...
```

Dla każdego połączenia serwera z hostem ustawiliśmy:

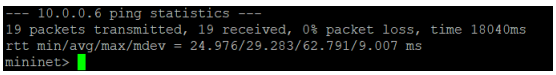
- delay = 0,1 ms,
- bw = 100 (100 Mb/s).

Dla każdego połączenia między różnymi serwerami ustawiliśmy:

- delay - zgodny z naszą topologią sieci,
- bw = 100 (100 Mb/s),
- loss - losowa liczba z przedziału (0,0; 6,0),
- max_queue_size = 2000.

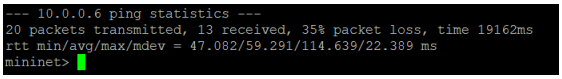
3 Zadanie 2

Na samym początku użyliśmy komendy ping w relacji sieci h4 i h6. W celu sprawdzenia, czy nasza relacja w przybliżeniu daje nam wartości realne czasu RTT, sprawdziliśmy ile powinien wynieść czas teoretyczny między tymi hostami. Z obliczeń wynika, że jego wartość powinna wynosić dwukrotnej sumie opóźnień między hostami, czyli około 22,4 ms. Po wykonaniu komendy ping byliśmy w stanie faktycznie zobaczyć, że te opóźnienie jest zbliżone do wartości teoretycznej.



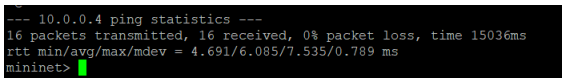
Rysunek 3: komenda h4 ping h6

Aby przetestować, czy zmiana delay'u powoduje zmianę czasu RTT, postanowliśmy zwiększyć dwukrotnie opóźnienie na drogach, które wchodziły w skład połączenia pomiędzy h4 i h6. Z naszego eksperymentu wyszło, że dwukrotne zwiększenie tej charakterystyki łączy wpływa na zmianę parametru przesyłu danych, jakim jest RTT, podwajając jego wartość.



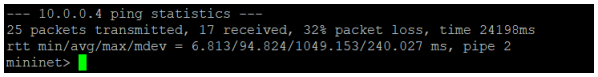
Rysunek 4: komenda h4 ping h6 przy dwukrotnym delay'u

Następnym naszym doświadczeniem było zbadanie wpływu wartości straty pakietów na parametry przesyłu danych. Na samym początku ustawiliśmy wartość loss pomiędzy Toulouse, a Barceloną na 0,5 (czyli prawdopodobieństwo utraty pakietów wynosiło 0,5%). Następnie użyliśmy komendy h2 ping h4, aby sprawdzić statystyki przesyłu danych.



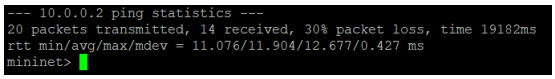
Rysunek 5: komenda h2 ping h4 przy loss 0.5

Po krótkiej analizie danych, zmieniliśmy wartość straty pakietu danych na 30%. Jak można zauważyć na rysunku poniżej loss ma wpływ nie tylko na liczbę straconych pakietów, ale również na czas RTT.



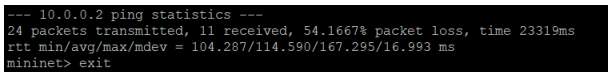
Rysunek 6: komenda h2 ping h4 przy loss 30

Ostatnim doświadczeniem wykonanym przez nas była zmiana wielkości przepustowości łącza. Wiedząc o tym, że Mininet ma ograniczenie do obsługi przepustowości do wartośći 1Gbs ustawiliśmy wartość bw między h1 i h2 na wartość 100, czyli 100 Mbs oraz zwiększyliśmy rozmiar wysyłanych danych do 8000B, ponieważ dla domyślnej wartości czas RTT nie ulegał znaczącej zmianie.



Rysunek 7: komenda h1 ping h2 przy bw 100

Następnie zmniejszyliśmy wartość przepustowości łącza 100 krotnie na wartość 1, czyli 1Mbs. Jak widać na rysunku poniżej, wartość RTT uległa znaczącej zmianie, zwiekszając się ponad 10 krotnie.



Rysunek 8: komenda h1 ping h2 przy bw 1

3.1 Wnioski

Z powyższych doświadczeń można jasno wywnioskować, że zmiana charakterystyki łączy ma znaczący wpływ na wartości parametrów przesyłu danych.

- **delay** - zwiekszając delay, RTT będzie rosło dwukrotnie, w zależności od tego, na jakiej drodze zostanie zwiększone opóźnienie.
- **loss** - zwiekszając loss prawdopodobieństwo utraty większej ilości pakietów będzie rosło, co wpłynie znacząco na czas RTT, jak udało nam się to zobrazować.
- **bw** - zmniejszając przepustowość łącza, czas RTT zwiększy się, gdyż zmniejszona przepustowość skutkuje nagromadzeniem danych w kolejkach węzłów pośrednich, co z kolei prowadzi do opóźnień w dostarczaniu danych na cel.

4 Zadanie 3

Przy użycie komendy w Mininetcie xterm h2 h7 otworzyliśmy terminale graficzne przy pomocy Xming. Następnie w terminalu, który należy do h2 użyliśmy komendy

```
python -m http.server 80 &
```

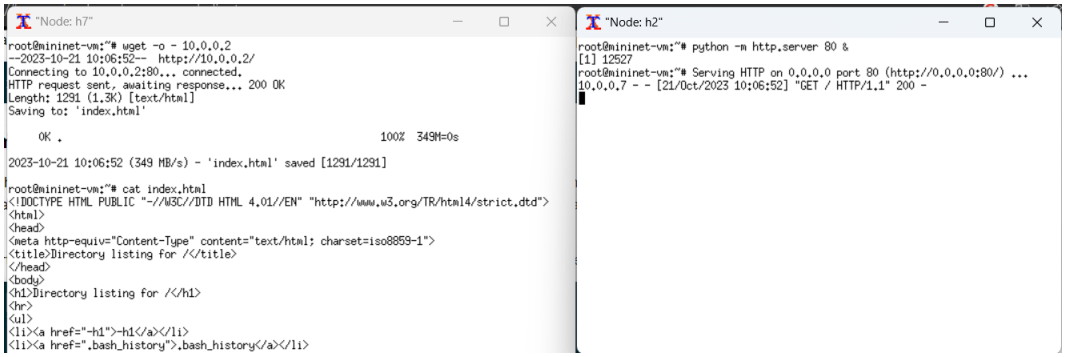
aby postawić serwer na h2, a w drugim terminalu użyliśmy

```
wget -O - 10.0.0.2
```

aby pobrać dane z serwera postawionego na tym hoście. Następnie użyliśmy komendy

```
cat index.html
```

aby sprawdzić, czy plik został rzeczywiście pobrany na naszym serwerze.



Rysunek 9: Terminale h2 i h7 po użyciu komend

Z powyższego rysunku wynika to, że plik został prawidłowo pobrany, co doprowadziło nas do konkluzji, że wszelkie działania terminali oraz łączenia między klientem i serwerem były udane.

5 Zadanie 4

Kolejnym zadaniem, jakie musieliśmy zrobić było zbadanie jak wartości otrzymywanych parametrów przesyłu danych zależą od typu i parametrów generowanego strumienia danych oraz od drogi przesyłu danych i charakterystyki należących do niej łączy. Aby nie gromadzić dużej ilości screenów, postanowliśmy stworzyć tabele, w których pokażemy podsumowanie doświadczenia przy zmianie parametrów.

5.1 UDP

5.1.1 Parametr block_rate

Pierwszym parametrem, który postanowiliśmy zbadać był parametr -b, który jest używany do określenia ilości pakietów na sekundę (pps). Poniżej na rysunku znajduje się tabela podsumowująca testy dla parametru -b z wybranymi wartościami (testów przeprowadziliśmy więcej, ale zostało to zawarte we wnioskach), gdy ten wynosił odpowiednio: 1500, 1200, 700 między h6 i h9.

badanie wpływu parametru -b

	Transfer (Mbytes)	bandwith (Mbits/s)	jitter (ms)	lost (%)	Latency avg (ms)	pps	NetPwr
większy niż przepustowość (-b 1500)	24,8	9,59	0,45	13	1338,729	1199	0,9
równy przepustwości (-b 1200)	22,7	9,52	0,317	0,92	5,65	1189	210,55
mniejszy niż przepustowość (-b 700)	13,2	5,54	1,126	0,99	3,67	692	188,81

Rysunek 10: Tabela testów dla parametru -b

Wnioski

- **Gdy parametr "block rate"był mniejszy od domyślnej wartości** - zgodnie z naszymi oczekiwaniami, nie zaobserwowaliśmy żadnych strat w przepustowości. Zauważyliśmy również, że wzrost ilości pakietów wysyłanych na sekundę (pps) wiązał się z zwiększeniem parametru "bandwidth". Przykładowo dla 700pps wyniósł on 5,54 Mb/s, a dla 1000pps 7,93 Mb/s. To zrozumiałe, ponieważ wysyłając więcej pakietów, jesteśmy w stanie przesłać większą ilość danych, co naturalnie prowadzi do wykorzystania większej dostępnej przepustowości.
- **Gdy parametr "block rate"był równy domyślnej wartości** - na początku obserwowaliśmy niewielkie straty, wynoszące około 0,92% . Przy tym ustawieniu osiągnęliśmy przepustowość na poziomie 9,52 Mb/s . Aby jak uzyskać jak najmniejsze straty, doszliśmy do wniosku, że ustawiona przez nas przepustowość powinna być mniejsza od przepustowości domyślnej.
- **Gdy parametr "block rate"był większy od domyślnej wartości** - po tym doświadczeniu byliśmy w stanie wywnioskować, że wraz ze wzrostem parametru "b"straty pakietów wzrastały. Dla wartości 1500 wynoisły one 13%, a dla wartości 3000 - 57%.

W każdym z przeprowadzonych przez nas doświadczeń, proporcjonalnie do wzrostu parametru "b"wzrastała przepustwość sieci do momentu, w którym osiągnęła ona domyślną wartość.

5.1.2 Parametr wielkości bloku

Kolejnym parametrem, który postanowiliśmy zbadać był parametr "l", który jest używany do określenia wielkości bloku. Poniżej zamieściliśmy tabela podsumowującą testy dla parametru -l, gdy ten wynosił odpowiednio: 1000, 500, 100 i 20 między h6 i h9.

badanie wpływu parametru -l

	Transfer (Mbytes)	bandwith (Mbits/s)	jitter (ms)	lost (%)	Latency avg (ms)	pps	NetPwr
-l 1000	22,7	9,52	0,317	0,92	5,65	1189	210,55
-l 500	11,3	4,76	0,606	0,88	3,546	1190	167,8
-l 100	2,27	0,952	0,462	0,86	3,628	1189	32,79
-l 20	0,465	0,19	0,789	0,9	3,621	1189	6,57

Rysunek 11: Tabela testów dla parametru -l

Wnioski

- **Wydajność i przepustowość** - większy rozmiar bloku może wpłynąć na wydajność transmisji, ponieważ mniej pakietów jest generowanych do przesyłania tych samych danych. W przypadku większego rozmiaru bloku można osiągnąć większą przepustowość, ponieważ mniej nagłówków pakietów jest wymaganych w porównaniu do przesyłania tych samych danych w mniejszych blokach.
- **Straty i fragmentacja** - zbyt duży rozmiar bloku może prowadzić do problemów związanych z fragmentacją, szczególnie w sieciach z ograniczeniami dotyczącymi maksymalnego rozmiaru pakietu. Jeśli blok jest większy niż maksymalny rozmiar pakietu w sieci, może być konieczna fragmentacja bloku na mniejsze fragmenty. To może zwiększyć ryzyko utraty fragmentów, co wpłynie negatywnie na jakość transmisji. Dla -l = 2000 mieliśmy problem z uzyskaniem statystyk, ponieważ część datagramów nie została odebrana przez serwer.
- **Opóźnienia** - większy rozmiar bloku może spowodować zwiększone opóźnienia, ponieważ większe bloki mogą wymagać więcej czasu na przesłanie i przetworzenie, co może wpłynąć na czas dostarczenia danych do odbiorcy. Na przykład opóźnienie dla -l = 1000 było ponad 1,5 raza większa od opóźnień dla -l = 500.

5.2 TCP

5.2.1 Parametr max_bit_rate

Pierwszym parametrem, który postanowliśmy zbadać był -b, który definiuje prędkości przesyłu danych podczas przeprowadzania testu prędkości transmisji TCP. Poniżej na rysunku znajduje się tabela podsumowująca testy dla parametru -b z wybranymi wartościami, gdy ten wynosił odpowiednio: 10M, 5M, 4M, 1M, 0,5M między h4 i h2.

badanie wpływu parametru -b

	Transfer (Mbytes)	bandwith (Mbits/s)	Reads	czas trwania (s)
-b 10M	10	3,6	8260	23,321
-b 5M	10	3,71	8391	22,591
-b 4M	10	3,17	9165	26,461
-b 1M	10	1,05	46447	79,851
-b 0,5M	10	0,524	68014	160,002

Rysunek 12: Tabela testów dla parametru -b

Wnioski

- **Prędkość przesyłu danych** - głównym efektem zmiany parametru -b jest kontrola prędkości przesyłu danych między klientem a serwerem podczas testu. Gdy parametr "b"był mniejszy od wartości domyślnej przepustowość malała. Jest to zrozumiałe, ponieważ kontrolowaliśmy mniejszą prędkość przesyłu danych podczas testu. W rezultacie spadek prędkości przesyłu danych może prowadzić do zmniejszenia przepustowości (bandwidth). Jeżeli chodzi o badane wartości, np. 10M to bandwith nie rósł, ponieważ napotkał on ograniczenia sieciowe. Anologicznie jest z czasem trwania przesyłu danych.

5.2.2 Parametr volume

Kolejnym parametrem, który poddaliśmy testom był -n, służy do określenia ilości danych, która ma być przesłana podczas testu prędkości transmisji TCP. Poniżej na rysunku znajduje się tabela podsumowująca testy dla parametru -n z wybranymi wartościami, gdy ten wynosił odpowiednio: 50M, 10M, 5M, 1M, 0,5M między h4 i h2.

badanie wpływu parametru -n

	Transfer (Mbytes)	bandwith (Mbits/s)	Reads	czas trwania (s)
-n 50M	50	3,53	40755	118,94
-n 10M	10	3,6	8260	23,321
-n 5M	5	2,98	4318	14,069
-n 1M	1	2,89	875	2,903
-n 0,5M	0,5	3,85	578	1,09

Rysunek 13: Tabela testów dla parametru -n

Wnioski

- **Ilość danych do przesłania** - zmiana wartości parametru -n wpływa na ilość danych, która zostanie przesłana. Im większa wartość parametru -n, tym więcej danych zostanie przesłanych, a test potrwa dłużej. Zmniejszenie wartości -n spowoduje przesłanie mniejszej ilości danych, a test będzie krótszy.
- **Czas trwania testu** - zmiana ilości danych za pomocą parametru -n wpływa na czas trwania testu. Większa ilość danych wymaga dłuższego czasu na przesłanie, podczas gdy mniejsza ilość danych skraca czas trwania testu, co jest w pełni zrozumiałe.

5.2.3 Parametr block_size

Następnym parametrem, który zbadaliśmy był -l, który służy do określenia rozmiaru bloków danych (lub segmentów) używanych podczas testu prędkości transmisji TCP. Poniżej na rysunku znajduje się tabela podsumowująca testy dla parametru -l z wybranymi wartościami, gdy ten wynosił odpowiednio: 1, 2, 3, 4, 5 między h4 i h2.

badanie wpływu parametru -l

	Transfer (Mbytes)	bandwith (Mbits/s)	Reads	czas trwania (s)
-l 5	10	9,25	6364	9,07
-l 4	10	7,95	6284	10,56
-l 3	10	6,32	5915	13,27
-l 2	10	6,27	6932	13,372
-l 1	10	3,6	8260	23,321

Rysunek 14: Tabela testów dla parametru -l

Wnioski

- **Wydajność transmisji** - rozmiar bloków danych ma wpływ na wydajność transmisji TCP. Większe bloki danych mogą zwiększyć przepustowość, ponieważ mniej narzutu związanego z nagłówkami jest potrzebne do przesyłania danych. Również pozwala to na skrócenie czasu trwania transmisji, jak widać to między -l 1 i -l 2. Powyżej wartości -l 5 zarówno czas, jak i przepustowość nie zmieniały się znacząco.

5.2.4 Parametr window

Ostatnim parametrem, który zbadaliśmy był -w, który służy do określenia rozmiaru bufora przesyłu danych używanego podczas testu prędkości transmisji TCP. Poniżej na rysunku znajduje się tabela podsumowująca testy dla parametru -w z wybranymi wartościami, gdy ten wynosił odpowiednio: 100000, 50000, 25000, 10000, 5000 między h4 i h2.

badanie wpływu parametru -l

	Transfer (Mbytes)	bandwith (Mbits/s)	Reads	czas trwania (s)
-w 100000	10	3,53	8520	23,759
-w 50000	10	3,6	8260	23,321
-w 25000	10	3,54	8285	23,679
-w 10000	10	3,05	9622	27,528
-w 5000	10	0,885	25675	94,796

Rysunek 15: Tabela testów dla parametru -w

Wnioski

- **Wydajność przepustowości** - zwiększając rozmiar bufora, można zwiększyć przepustowość transmisji TCP. To może pomóc w uzyskaniu wyższej wydajności, szczególnie na łączach o długich trasach lub z dużym opóźnieniem.
- **Kontrola nad narzutem TCP** - rozmiar bufora wpływa na narzut związany z protokołem TCP, który może wpływać na wydajność. Większy bufor może pomóc w zmniejszeniu narzutu związanego z nagłówkami TCP, co może poprawić przepustowość oraz czas trwania transmisji.

6 Zadanie 5

6.1 UDP

6.1.1 Parametr block_rate

Badając relację między h3 (serwerem) a h8 (klientem) o przepustowości 10 Mbps, zauważyliśmy, że przy równoczesnym uruchomieniu dwóch połączeń, których suma przepustowości nie przekraczała 10 Mb/s nie występowały straty ani znaczne zmiany w przepustowości. Również tyczy się to sytuacji, w której odpaliliśmy 3 równoległe połączenia na takich samych zasadach.
Jednak kiedy zmieniliśmy wartości -b na takie, których suma przekroczyła 10 Mbps (w naszym teście 5 Mb/s + 3 Mb/s + 10 Mb/s) i uruchomiliśmy połączenia w tej kolejności, zauważyliśmy, że w pierwszych sekundach testu nie

występowały straty pakietów, a połączenia dzieliły się przepustowością, zgodnie z ich parametrem "block_rate", który był im przypisany w skrypcie (tzn. 1 połączenie uzyskało około $\frac{5}{18}$ maksymalnej przepustowości, 2 połączenie $\frac{3}{18}$, a 3 połączenie $\frac{10}{18}$ z 10 Mb/s). Po kilku sekundach zaczęliśmy doświadczać strat w ruchu sieciowym, a połączenie o najmniejszej przepustowości miało najmniejsze straty (33%), połączenie o średniej przepustowości miało straty na poziomie 43%, a połączenie o największej przepustowości miało straty na poziomie 52%. To zjawisko różniło się w zależności od kolejności uruchomienia połączeń i nie było spójne w każdym powtórzeniu eksperymentu.
Te wyniki sugerują, że rywalizacja o przepustowość między połączeniami była wpływana przez różne czynniki, takie jak kolejność uruchamiania, dynamiczne zmiany w sieci itp.

6.1.2 Parametr wielkości bloku

W kolejnym teście postanowliśmy zbadać wpływ parametru -l na końcowe statystyki. Poniżej zamieścliśmy tabelę z danymi, które przedstawiają dwa równoległe połączenia, każde o przepustowości 5 Mb/s, jedynie różniły się one rozmiarem wielkości: 1 połączenie - 5000B, a 2 - 100B.

badanie wpływu parametru -l

	Transfer (Mbytes)	bandwith (Mbits/s)	jitter (ms)	lost (%)	Latency avg (ms)	pps	NetPwr
1 połączenie (5000B)	6,08	2,66	0,53	2,8	5026,895	3331	0,07
2 połączenie (100B)	1,63	0,71	5,561	12	5022,421	60	0,02

Rysunek 16: Tabela testów dla parametru -l

Podczas eksperymentu z równoczesnym wykonywaniem dwóch połączeń ze zmienionym parametrem wielkości bloku, zauważyliśmy, że to połączenie, które generowało mniejszą ilość pakietów w krótszym czasie, było bardziej podatne na utratę pakietów. W rezultacie połączenie to doświadczało wyższej utraty pakietów - 12% w porównaniu do równocześnie działającego pierwszego połączenia, które miało 2,8% strat pakietów. Co ciekawe, ta niższa utrata wpłynęła również na przepustowość pierwszego połączenia, które w efekcie okazało się znacznie wyższe - 2,66 Mb/s niż na te, które miało mniejsze parametry wielkości bloku danych - 0,71 Mb/s.

6.2 TCP

6.2.1 Parametr sesji

Na samym początku postanowiliśmy zbadać parametr -P, który odpowiada za zwiększenie ilości sesji dla równoległych połączeń między h2 i h5. Dla podstawowego połączenia, bez zmiany parametru P, otrzymaliśmy:

badanie wpływu parametru -P

	bandwith (Mbits/s)	czas (s)
-P 1	2,95	28,457
-P 2	1 - 1,72 | 2 - 1,65	50,719
-P 4	1 - 0,89 | 2 - 0,87 | 3 - 0,87 | 4 - 0,87	96,269

Rysunek 17: Tabela testów dla parametru -P

Wywnioskowaliśmy, że w protokole TCP równomierne dzielenie przepustowości między połączeniami umożliwia każdemu połączeniu korzystanie z dostępnego pasma w uczciwy sposób, co jest ważne w przypadku wielu jednoczesnych użytkowników lub aplikacji.

6.2.2 Parametr volume

Następnym parametrem, który poddaliśmy testom był parametr volume. W skrypcie odpaliliśmy 3 równoczesne połączenia, o takich samych parametrach, jedynie zmieniliśmy wartości ilości danych na wartości: 1 połączenie - 3MB, 2 połączenie - 5MB, 3 połączenie - 10MB. Wnioski jakie uzyskaliśmy były następujące:

- **Różne ilości danych do przesłania** - każda sesja miała inną ilość danych do przesłania, zgodnie z wartością parametru -n w skrypcie.

- **Równoczesne przesłanie danych** - wszystkie sesje działały równocześnie i próbowały przesłać swoje dane w miarę możliwości.
- **Zróżnicowane czasy zakończenia** - ponieważ sesje przesyłają różne ilości danych, zakończyły się w różnych momentach. Sesje, które przesyłały mniejszą ilość danych, zakończyły się wcześniej niż te, które przesyłały większą ilość danych. W 1 połączeniu, które miało do przesłania 3 MB skończyło się w 9,72 s, 2 w 15,22 s, a 3 w 33,58 s.
- **Kontrola przepustowości** - pomimo różnych ilości danych, przepustowość sieci pozostały takie same. .

6.2.3 Parametr window

W kolejnym teście postanowiliśmy sprawdzić wpływ rozmiaru okna na charakterystyki łączy. W pierwszym teście ustawiliśmy przy 3 połączeniach ustawliliśmy rozmiary okna na wartości: 40000, 25000, 10000. Dla okna, które było najbliżej wartości naszego optymalnego okna jego przepustowość miała największą wartość równą 2,60 Mb/s, dla okna drugiego w kolejności 2,39 Mb/s, a dla ostatniego 1,89 Mb/s.
W kolejnym teście odpaliliśmy 2 równoległe połączenia z bardziej skrajnymi wartościami rozmiaru okna: 40000 i 400, aby sprawdzić, czy zmiany przepustowości będą jeszcze większe. Zgodnie z naszyi przypuszczeniami, gdy rozmiar okna był większy, bandwith wynosił 2,71 Mb/s, a gdy mniejszy 0,824 Mb/s. Wiemy jednak, że rozmiar okna TCP kontroluje, ile danych może być przesłane między nadawcą a odbiorcą przed otrzymaniem potwierdzenia. Logiczne jest więc, że małe okno oznacza, że nadawca może wysłać tylko niewielką ilość danych, zanim musi czekać na potwierdzenie. To ogranicza efektywność przesyłania danych.

6.3 UDP a TCP

Ostatecznym testem, do jakiego podeszliśmy było uruchomienie równoległego połączenia TCP i UDP na tym samym łączu, w naszym przypadku na h1-h10. Na początku ustawiliśmy jednakowe parametry w obu połączeniach zmieniając jedynie parametr wielkości bloku. Dla małej wartości parametru -l przez czas działania UDP, przepustowść TCP była bliska 0 Mb/s, a w niektórych momentach nawet tyle wynosiła. Po zakończeniu działania UDP, protkól TCP zaczął normalnie działać. Dla większego parametru -l TCP odnosiło znacznie mniejsze straty pakietów. Dzieje się tak, gdyż podzas gdy UDP działa z dużą częstotliwością i małym rozmiarem paczek, może zdominować łącze sieciowe, zapchać bufor i spowodować, że protokół TCP nie jest w stanie efektywnie przesyłać danych. Dopiero po zakończeniu działania UDP, TCP może normalnie działać, ponieważ nie ma już konkurencji z dużą ilością paczek UDP. Jednak przy większym rozmiarze paczki w TCP, protokół ten jest w stanie przesyłać więcej danych w jednej paczce, co obniża narzut na kontrolę przepływu i zwiększa efektywność przesyłania.

7 Podsumowanie

W ramach przeprowadzonego laboratorium mieliśmy okazję eksperymentować z tworzeniem płaszczyzny danych sieciowej oraz testować wydajność połączeń między hostami za pomocą narzędzia iperf, zarówno przy użyciu protokołu TCP, jak i UDP. To doświadczenie pozwoliło nam lepiej zrozumieć i ocenić różnice między tymi dwoma protokołami.

Testy za pomocą TCP i UDP pozwoliły nam zbadać, jak zmiana parametrów oraz charakterystyki łączy może wpłynać na parametry przesyłu danych. Dowiedzieliśmy się, że TCP jest bardziej sprawiedliwy od UDP, ponieważ zapewnia kontrolę nad przepływem danych i gwarantuje niezawodność poprzez retransmisję pakietów w przypadku utraty. UDP jest mniej sprawiedliwy, ponieważ nie zapewnia tych mechanizmów i może prowadzić do nierównowagi w dostępie do zasobów sieciowych, ale ostateczne użycie odpowiedniego protokołu zależy od konkretnych wymagań aplikacji.

Laboratorium dostarczyło nam wartościowej wiedzy na temat testowania wydajności sieci przy użyciu różnych protokołów, co może okazać się niezwykle przydatne w przyszł pracy w dziedzinie cyberbezpieczeństwa.