Program został przetestowany za pomocą połączonych ze sobą maszyn wirtualnych. Do analizy prędkości i ilości przesyłanych danych wykorzystano program Wireshark. Przepustowość łącza, opóźnienie i procent zagubionych pakietów były emulowane za pomocą modułu netem i, o ile nie jest napisane w inaczej w opisie testu, ustawione są domyślnie.

Wyniki testów są podawane w następujących jednostkach:
czas komunikacji: sekundy | ilość przesłanych danych: bajty | średnia prędkość: bajty na sekundę

Test wydajności protokołów w warunkach sieciowych o niskim opóźnieniu i niskim procencie utraconych pakietów.

Przetestujemy przesyłanie małej wielkości ciągów. Konfiguracja testu:

- opóźnienie: 100 ms u serwera i klienta
- maksymalna wielkość paczki: 64000B
- rozmiar pliku: zmienna w tabeli

rozmiar pliku: 1000B	TCP	UDP	UDPR	10000B	TCP	UDP	UDPR
Czas komunikacji	0.75	0.32	0.4		0.7	0.36	0.3
Ilość przesłanych danych	1865	1225	1225		11129	10429	10429
Średnia prędkość	2500	3831	3070		15791	29106	34350
Liczba pakietów	12	4	4		16	10	10

Obserwacje i wnioski z testów:

Protokoły UDP oraz UDPR są szybsze niż TCP w warunkach sieciowych o niskim opóźnieniu i niskim procencie utraconych pakietów, gdy przesyłamy niewielkie wiadomości. Wynika to z faktu, że protokół TCP musi przeprowadzić fazę negocjacji przed rozpoczęciem wysyłania danych, co może mieć istotny wpływ na całkowity czas transmisji danych.

Przetestujemy przesyłanie średniej wielkości danych i wpływ maksymalnej wielkość paczki DATA. Konfiguracja testu:

- opóźnienia: 100 ms u serwera i klienta
- rozmiar pliku: 250000B
- maksymalna wielkość paczki DATA: zmienna w tabeli

64000	TCP	UDP	UDPR	10000	TCP	UDP	UDPR	1000	TCP	UDP	UDPR
Czas	1.52	0.34	0.95		2.04	0.47	5.52		1.53	0.38	51.17
Ilość danych	260564	256126	256303		260477	257052	252103		265268	265912	280603
Średnia prędkość	171751	762631	271010		127816	271010	45639		173818	698886	5483
Liczba pakietów	158	175	178		152	178	206		152	253	508

Wysyłanie ciągu protokołem UDP zakończyło się sukcesem (maksymalna wielkość paczki DATA: ile sukcesów na ile testów):
64000B: 10 na 10 razy | 10000B: 1 na 20 razy | 1000B: 4 na 10 razy

Obserwacje i wnioski z testów:

Przy przesyłaniu danych o średniej wielkości protokół UDP przestaje gwarantować niezawodność przesyłania ciągu danych, ponieważ nie ma żadnego mechanizmu retransmisji utraconych pakietów lub sortowania paczek w przypadku odebrania ich w złej kolejności. Niemniej jednak, jeśli te problemy nie wystąpią, protokół UDP osiąga zauważalnie wyższą prędkość niż TCP.

Protokół UDPR wydaje się być zaprojektowany w celu rozwiązania tych problemów, łącząc szybkość protokołu UDP z niezawodnością przesyłania danych.

Jednak, w miarę wzrostu liczby wysyłanych pakietów, które wymagają potwierdzenia, obserwujemy znaczne obniżenie prędkości przesyłania. Natomiast okazuje się, że protokół TCP, pomimo skomplikowanego mechanizmu potwierdzania pakietów, jest w stanie zachować podobną wydajność bez względu na rozmiar paczek, więc w tym przypadku jest lepszym wyborem.

Przetestujemy przesyłanie dużej wielkości danych i wpływ przepustowości. Konfiguracja testu:

- opóźnienia: 25 ms u serwera i klienta
- rozmiar pliku: 10^8B
- maksymalna wielkość paczki: 64000B
- przepustowość: zmienna w tabeli

10 Mb/s	TCP	UDPR	100 Mb/s	TCP	UDPR

Czas komunikacji	83.87	100.77		19.18	36.03
Ilość przesłanych danych	100395461	102475147		100349519	102475147
Średnia prędkość	1197067	1016995		5231833	2844180
Liczba pakietów	5487	70329		4790	70321

Obserwacje i wnioski z testów:

Protokół UDP nie nadaje się do przesyłania plików o tak dużej wielkości, ze względu na brak gwarancji dostarczenia danych.

Zarówno protokół TCP, jak i UDPR, były ograniczane przez przepustowość łącza podczas pierwszego testu, więc nie dostrzegamy dużej różnicy w wydajności, natomiast w drugim teście widzimy, że średnia prędkość protokołu TCP jest dwukrotnie wyższa niż protokołu UDPR. Zatem, w przypadku przesyłania dużych plików, protokół TCP jest preferowanym wyborem.

Przetestujemy zachowanie protokołu UDPR podczas ataku DDoS, zarówno na serwer, jak i na klienta komunikującego się z serwerem. Konfiguracja testu:

- opóźnienia: 250 ms u serwera i obu klienta
- rozmiar pliku: 10^7B
- maksymalna wielkość paczki: 32000B

	Bez ataku	Atak na serwer	Atak na klienta
Czas komunikacji	161.58	166.30	165.01
Ilość przesłanych danych	10261397	31667353	10261397
Średnia prędkość	63505	190412	62185
Liczba wysłanych pakietów	7206	22528	7210

Obserwacje i wnioski z testów:

Atak na serwer lub klienta nie wpływa znacząco na czas komunikacji. Mogłoby się wydawać, że jeśli co druga wymiana jest z innym klientem, to czas komunikacji powinien się dwukrotnie zwiększyć z powodu ustawionego opóźnienia.

Okazuje się jednak, że funkcja **sendto** nie blokuje się na okres opóźnienia, tylko umieszcza wiadomość do przesłania w buforze. Po upływie czasu opóźnienia, wiadomości do obu klientów są wysyłane w tym samym momencie. W rezultacie nie obserwujemy dwukrotnego zwiększenia czasu komunikacji.

Przetestujemy zachowanie protokołu UDPR przy utracie pakietów. Konfiguracja testów:

- maksymalna wielkość paczki: 64000B | rozmiar pliku: 5 * 10^6B | procent utraty pakietów: 5%
- maksymalna wielkość paczki: 1000B | rozmiar pliku: 5 * 10^4B | procent utraty pakietów: 5%
- maksymalna wielkość paczki: 1000B | rozmiar pliku: 2 * 10^4B | procent utraty pakietów: 25%

	TCP		TCP	UDPR		TCP	UDPR
Czas komunikacji	2.8		0.03	8.21		1.4	43.67
Ilość przesłanych danych	5088213		54726	56498		58000	71440
Średnia prędkość	1822270		2127474	6881		41431	1636
Liczba wysłanych pakietów	1231		54	107		59	137

Obserwacje i wnioski z testów:

W pierwszym teście protokół UDPR nie pozwala na skomunikowanie się. Spowodowane jest to tym, że pakiet o rozmiarze 64000B jest dzielony na mniejsze pakiety o wielkości 1514B. Utrata jednego z nich wystarczy, aby serwer nie odbierał danych poprawnie. Protokół TCP działa bez zarzutów.

W drugim teście zmniejszyliśmy maksymalny rozmiar paczki, tak żeby jedna paczka danych była wysyłana w jednym pakiecie. Mimo to, ze względu na ustawienie parametru MAX WAIT na 1 sekundę, każda utrata pakietu powodowała opóźnienie. W rezultacie protokół TCP okazał się zauważalnie szybszy.

W trzecim teście, mimo że protokół TCP wydaje się lepszy, to przesyłanie bajtów kończy się sukcesem tylko 2 na 10 razy, ponieważ utrata pakietów na poziomie 25% może spowodować urwanie się połączenia poprzez timeout. W przypadku protokołu UDPR, mimo dłuższego czasu działania, gwarantuje on przesłanie całego ciągu. Ma on jednak poważny problem, ponieważ

ostatni pakiet ACC nie jest retransmitowany i wówczas przy dużej utracie pakietów klient może go nie otrzymać i zakończyć się błędem.

Obserwujemy kolejną ciekawą zależność: w testach bez utraty pakietów protokół UDPR lepiej działał w zestawieniu z maksymalną wielkością paczki danych ustawioną na 64000B. Natomiast teraz widzimy, że lepiej sprawdza się mniejsza wielkość paczki danych.

Przetestujemy protokół UDPR w ekstremalnych warunkach opóźnienia.

- opóźnienia: 1000 ms u serwera i obu klienta
- MAX WAIT: 1s
- rozmiar pliku: 10^6B
- maksymalna wielkość paczki: 64000B

	UDPR
Czas komunikacji	33.4
Ilość przesłanych danych	2114808
Średnia prędkość	63311
Liczba wysłanych pakietów	1455

Obserwacje i wnioski z testów:

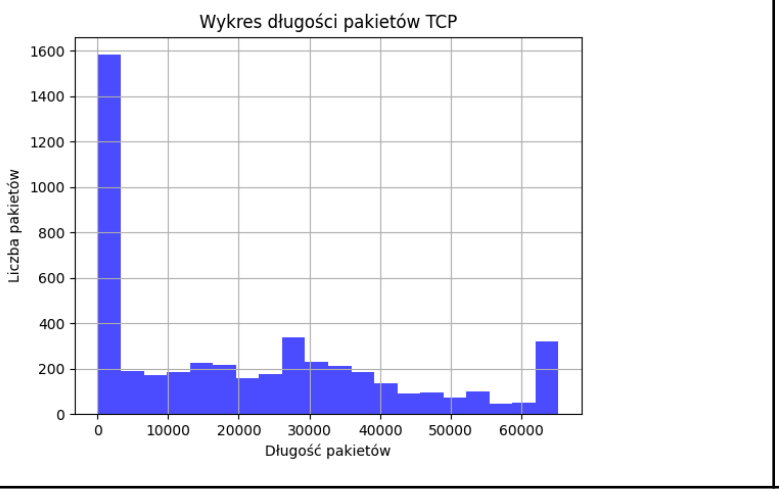
Nie licząc ostatniego potwierdzenia ACC oraz pakietu RCVD, przesyłanie danych przebiegło pomyślnie. Warto zauważyć, że protokół UDPR jest jedynym z wymienionych (w ramach tego zadania), który pozwala na przesłanie ciągu bajtów nawet w sytuacji, gdy opóźnienie przekracza maksymalny czas oczekiwania na wiadomość. Konsekwencją jest natomiast, błąd zakończenia u klienta.

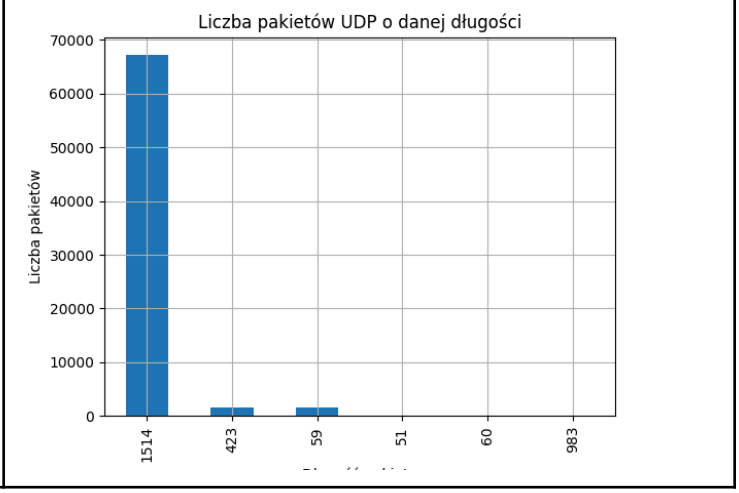
Podsumowanie:

1. Protokół TCP, mimo skomplikowanych mechanizmów zagwarantowania niezawodności, wypadł dobrze we wszystkich przeprowadzonych przez nas testach. Niezależnie od przepustowości łącza, utraty pakietów czy wielkości przesyłanego ciągu danych zachowuje wysoką prędkość przesyłu danych i gwarantuje poprawne przesłanie ciągu bajtów.
2. Protokół UDP sprawdza się bardzo dobrze, nawet lepiej niż protokół TCP, w przypadku przesyłania ciągów bajtów o małej wielkości. Rzeczywiście jest on wykorzystywany w takich celach, jak na przykład wysyłanie pakietów DNS. Jednakże, wraz ze wzrostem wielkości paczki, zaczyna mieć problemy wynikające z gwarancją dostarczenia wiadomości.
3. Protokół UDPR wydaje się być zaprojektowany w celu rozwiązania tych problemów, łącząc szybkość protokołu UDP z niezawodnością przesyłania danych. Niestety nie sprawdza się on najlepiej w przypadku przesyłanie dużej liczby pakietów. W przeciwieństwie do protokołu TCP, zauważalne jest duże opóźnienie wynikające z potwierdzeń.

W zależności od potrzeb i warunków sieciowych, każdy z tych protokołów może być odpowiedni do różnych zastosowań.

Spójrzmy jeszcze jak wyglądają długości pakietów, pomimo ustawienia programu, aby zawsze wysyłał maksymalną wielkość:





Wysyłając pakiety przy użyciu protokołu UDP, długość pakietów jest ograniczona (z reguły jest ograniczana do wielkości najmniejszego rozmiaru ramki na trasie pakietu, przez który przechodzi), podczas gdy w przypadku TCP wielkości pakietów są zróżnicowane, ponieważ dane są dzielone na segmenty o zmiennej wielkości.