Testy były przeprowadzone na dwóch maszynach wirtualnych i polegały na przesłaniu losowych danych od klienta do serwera. Środowisko: Linux Mint 21.3, jądro Linux 5.15.0-105-generic.

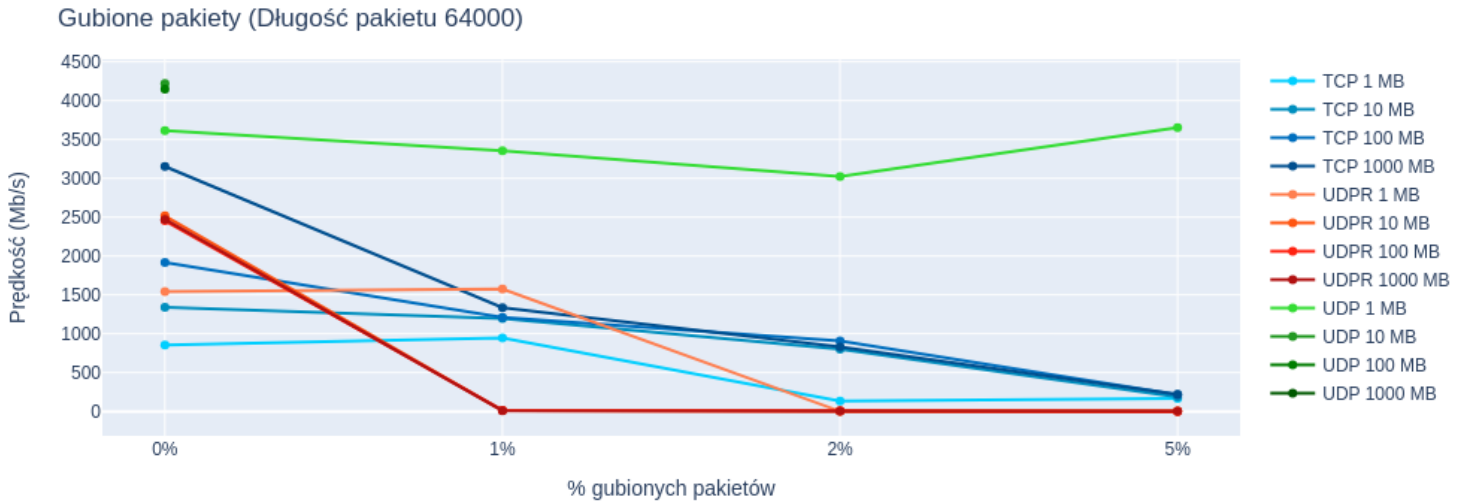
Podana na wykresach prędkość była liczona na podstawie rozmiaru danych i czasu, gdzie czas to okres od wysłania pakietu *CONN* (w przypadku tcp od wywołania funkcji *connect()*) przez klienta do odebrania pakietu *RECV*.
Czas wysłania/odebrania powyższych pakietów był sprawdzany za pomocą pola w kolumnie „Time” w Wireshark’u.
Celem było pominięcie czasu wczytywania danych ze standardowego wejścia przez klienta.

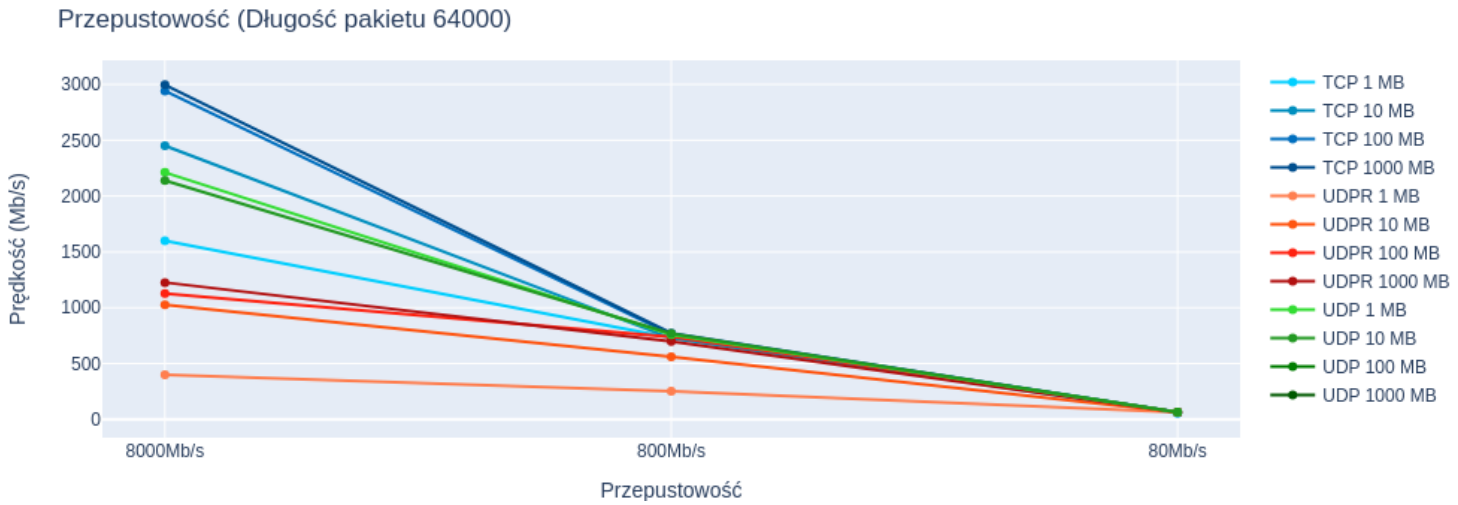
Podczas testów udpr stałe były równe MAX_WAIT = 1s, MAX_RETRANSMITS = 15, a dla testów udp/tcp MAX_WAIT = 1000s.

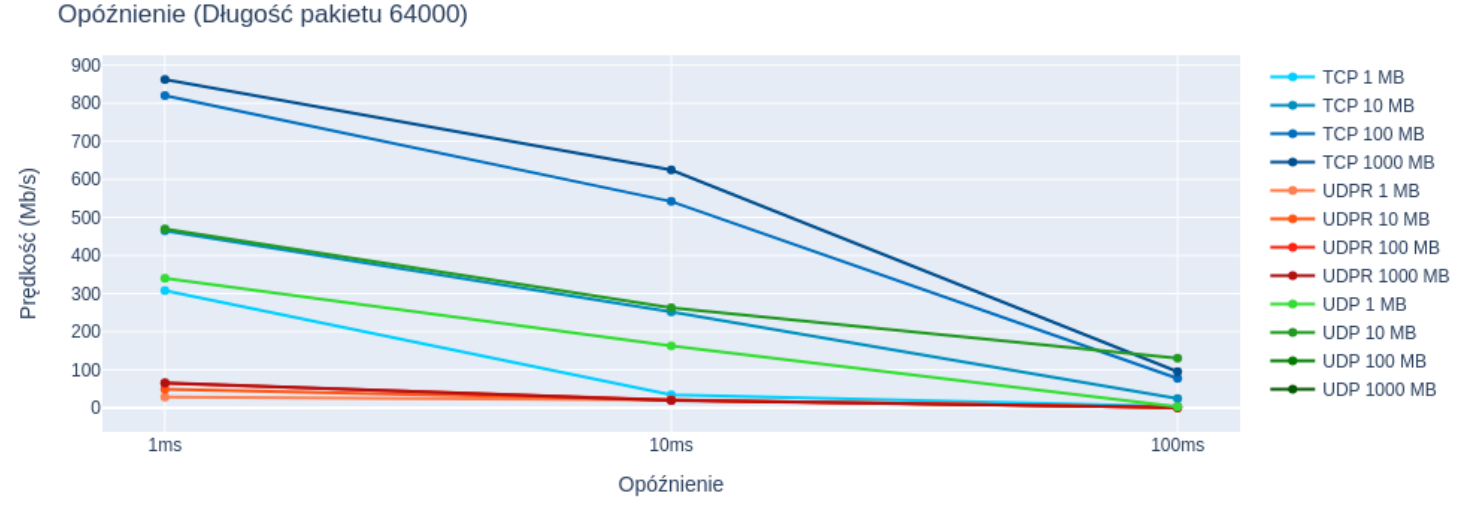
Każda wartość na wykresie jest średnią arytmetyczną z trzech pomiarów, dla których klient zakończył się sukcesem.
Brakujące wartości dla udp oznaczają, że pomimo kilku prób, nie udało się zebrać danych.

Wielkość obok nazwy protokołu na legendzie wykresu to rozmiar danych wejściowych.

1) Długość pakietu = 64000





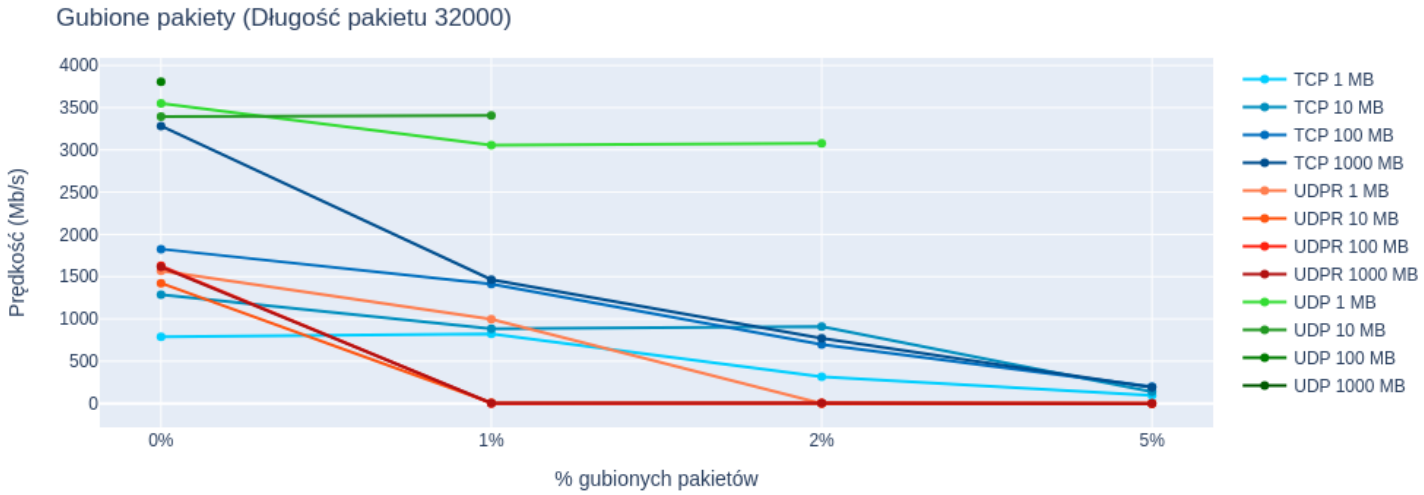


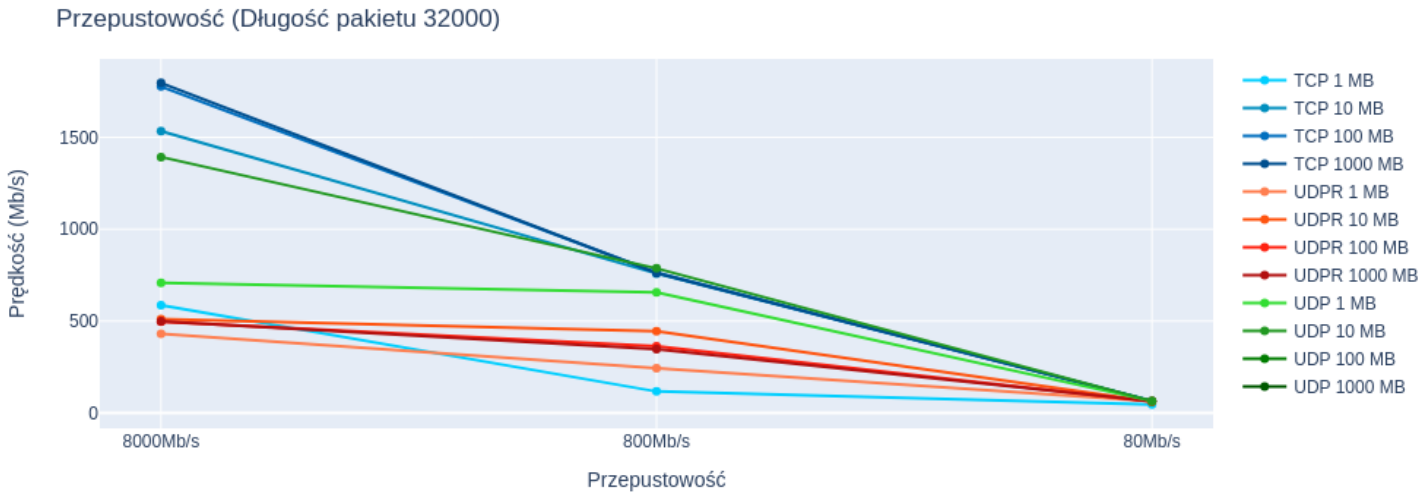
- Z pierwszego wykresu widać, że w przypadku udpr zaledwie 1% gubionych pakietów zmniejsza prędkość o dwa rzędy wielkości do około 10Mb/s, co wynika z długiego czasu oczekiwania na retransmisję (1s). Jedynie udpr-1MB nie stracił na prędkości dla 1% zagubionych pakietów, bo szczęśliwie żaden pakiet nie zaginął (to prawdopodobne, bo wysyłanych jest tylko 15 pakietów). Z tego samego powodu udp-1MB zachowało niezmienioną prędkość – żaden pakiet nie zaginął.
  Widać, że najlepiej zachowuje się tcp, które pomyślnie dostarcza wszystkie dane, tracąc najmniej (cały czas dużo) na wydajności.

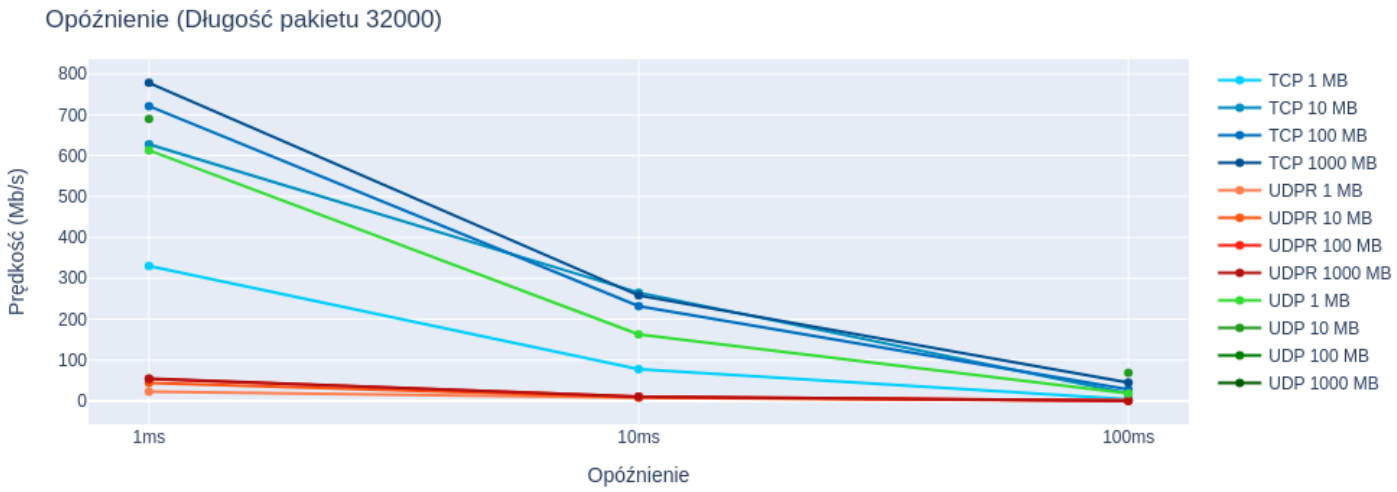
- Z drugiego wykresu wynika, że gdy przepustowość jest odpowiednio niska, różnice między protokołami zacierają się. Można podejrzewać, że nieznacznie większe wartości prędkości udp (dla przepustowości 800 i 80 Mb/s) są spowodowane mniejszym narzutem związanym z rozmiarem nagłówka.

- Z trzeciego wykresu widać, że nawet milisekundowe opóźnienie, ma bardzo negatywny wpływ na udpr (spowodowane tym, że udpr czeka na potwierdzenie każdej kolejnej wiadomości i jest to czas, który się marnuje). Tcp radzi sobie znacznie lepiej ze względu na potokowanie (wysyłanie wielu pakietów, a potwierdzenie dla każdego z nich może przyjść później i być zbiorcze). Udp ma największą wydajność, biorąc pod uwagę rozmiar pliku, bo czeka tylko na dwie wiadomości zwrotne – CONACC, RCVD.

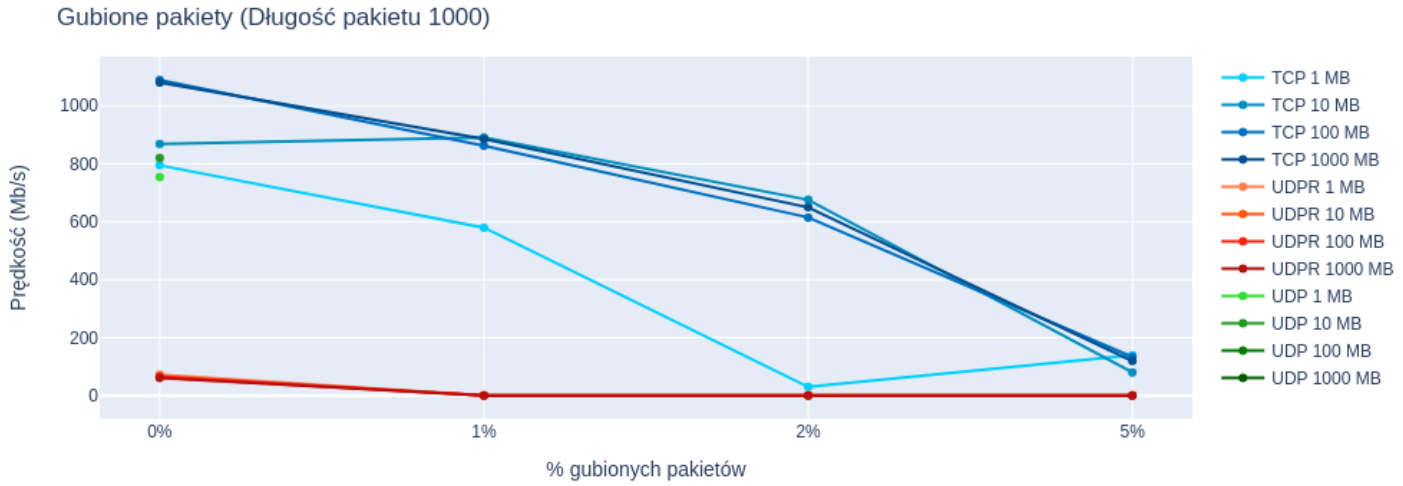
2) Długość pakietu = 32000

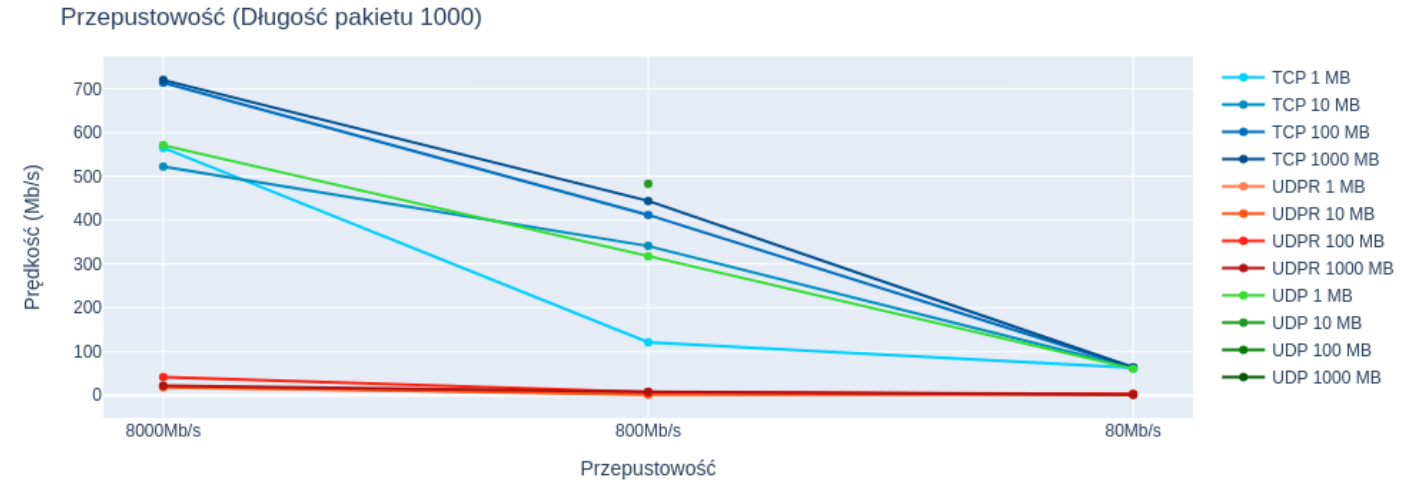


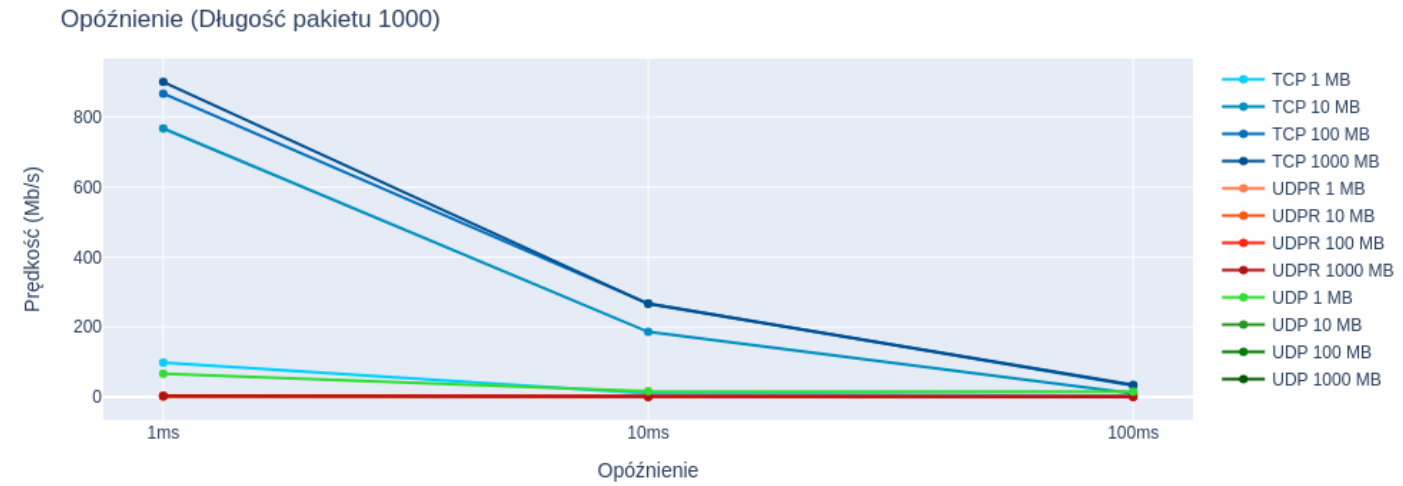




3) Długość pakietu = 1000







Patrząc na wszystkie wykresy widać, że udp jest zazwyczaj najszybsze, biorąc pod uwagę rozmiar pliku.
Tcp jest zazwyczaj szybsze od udpr ze zwiększającą się przewagą w miarę zmniejszania pakietów (więcej pakietów, to większa liczba potwierdzeń, na które udpr będzie czekało i tym wolniej będzie działać), dodatkowo tcp buforuje dane, czekając, aż uzbiera się większa porcja, co zmniejsza narzut związany z przesłaniem danych również w innych warstwach.

Wyraźnie widać również, że średnia prędkość przesyłania danych w ramach danego protokołu rośnie wraz ze wzrostem rozmiaru pliku, przy czym przyrost jest coraz mniejszy dla coraz większych danych. Jest to spowodowane narzutem na rozpoczęcie połączenia i jego zakończenie (pakiety CONN, CONACC, RECV), a w przypadku tcp dodatkowo „powolnym startem" ograniczającym początkowy rozmiar okna tcp.

Widać też, że udp często nie daje rady poprawnie wysłać dużej wiadomości. Konkretniej, dla długości pakietu równej 64000 oraz bez wprowadzania dodatkowych opóźnień, gubienia pakietów itd. 1MB udało się przesłać 98/100 razy, ale dane rozmiaru 100MB już tylko 8/100.
Modyfikując serwer, tak żeby ten odbierał wszystkie pakiety i korzystając z opcji Statistics->Conversations w Wireshark'u, okazuje się, że z wysłanych 100MB dociera średnio zaledwie 63MB (na podstawie dziesięciu prób).
Zmniejszenie długości przesyłanych pakietów na 1000 jedynie pogorszy sytuację, bo przesyłane dane będą dzielone na ramki rozmiaru ok. 1000B zamiast 1500B, co zwiększa liczbę danych, które mogą zaginąć, przy czym, jeśli którykolwiek fragment zaginie, to cały datagram również przepada. Udp gubi pakiety, nawet jeśli korzystamy z pętli zwrotnej, o czym łatwo można się przekonać, wysyłając dużo danych w krótkim czasie, przepełniając tym samym jeden z buforów systemowych.

Dla porównania, podczas testów, klient korzystający z dowolnego z pozostałych protokołów zawsze kończył się sukcesem.