

# SCHT - Laboratorium 1

Stanisław Ciszewicz (324 906), Jakub Kuszner (324 924)

Politechnika Warszawska, Cyberbezpieczeństwo

3 marca 2024

## Spis treści

<b>1. Zadanie 1</b>	2
1.1. Projektowanie sieci	2
<b>2. Zadanie 2</b>	3
2.1. Zależność między opóźnieniem a RTT	3
2.2. Badanie zależności pomiędzy RTT a przepustowością	4
2.3. Wnioski	5
<b>3. Zadanie 3</b>	5
<b>4. Zadanie 4</b>	6
4.1. TCP	6
4.1.1. Zależność między wielkością okna a przepustowością i RTT	6
4.1.2. Zależność między wielkością bufora a przepustowością	7
4.1.3. Badanie zmian przepustowości danych po dodaniu flagi -b	7
4.1.4. Zmiana przepustowości na jednym z łączy	7
4.1.5. Wnioski	7
4.2. UDP	8
4.2.1. Badanie block_rate na jakość transmisji	8
4.2.2. Badanie wpływu wielkości bloku na jakość transmisji	8
4.2.3. Zmiana przepustowości na jednym z łączy	8
4.2.4. Wnioski	8
<b>5. Zadanie 5</b>	9
5.1. TCP vs TCP	9
5.1.1. Badanie jakości połączenia przy kilku sesjach TCP przechodzących przez to samo łącze	9
5.1.2. Podział przepustowości w TCP dla ustawionych różnych wielkości okien	9
5.1.3. Ograniczenie przepustowości na jednym łączy	9
5.1.4. Wnioski	9
5.2. UDP vs UDP	9
5.2.1. Badanie jakości połączenia przy kilku sesjach UDP przechodzących przez to samo łącze	9
5.2.2. Zmniejszenie przepustowości na łączy, przez które przechodzi transmisja	10
5.2.3. Wnioski	10
5.3. UDP vs TCP	10
<b>6. Podsumowanie</b>	10

## 1. Zadanie 1

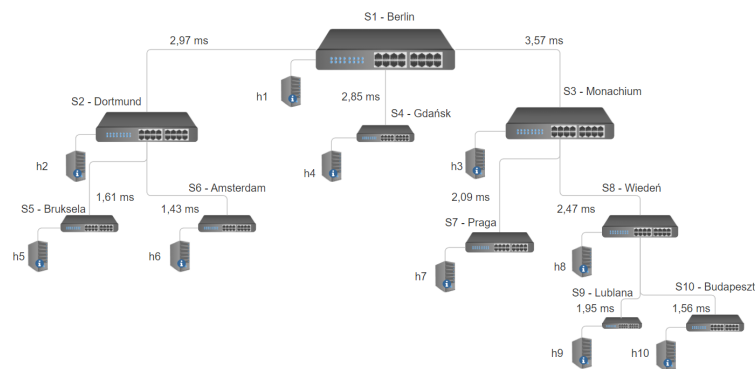
### 1.1. Projektowanie sieci

W pierwszym kroku przystąpiliśmy do napisania kodu w języku python implementującego prostą topologię trzech switchy wraz z jednym hostem dołączonym do każdego z nich. Mieliśmy tym samym sprawdzić, czy napisany przez nas kod działa poprawnie, zanim przystąpimy do projektowania docelowej, rozbudowanej sieci. Kod został napisany w oparciu o przykład zaprezentowany na zajęciach wykładowych. W naszym kodzie znajdują się 3 najważniejsze fragmenty:

- tworzenie switchy  
`s1 = self.addSwitch("s1") #Berlin`
- tworzenie hostów  
`h1 = self.addHost("h1") #Berlin`
- dodawanie połączeń między węzłami  
`self.addLink(h1, s1, delay='0.8ms', bw=100)`

Napisany kod w języku python przenieśliśmy na maszynę wirtualną Mininet aplikacją PuTTY wykorzystując komendę pscp. W kolejnym kroku przystąpiliśmy do przetestowania napisanego programu w środowisku Mininet - uruchomiliśmy naszą sieć komendą: `sudo mn -custom smallnetwork.py -topo mytopo -link tc`. Po utworzeniu sieci komendami kolejno: `nodes`, `links`, `pingall` dowiedliśmy, że utworzona przez nas sieć ma węzły i połączenia dokładnie takie, jakie zadeklarowaliśmy w kodzie oraz, że pakiet kontrolny przechodzi między każdą parą hostów.

Po uruchomieniu sieci testowej zaprojektowaliśmy docelową, składającą się z 10 dużych miast europejskich (w każdym po jednym switchu i jednym gościem). Dodatkowym założeniem był fakt, iż nasza struktura musiała mieć charakter drzewa tzn. nie mogły w niej występować żadne cykle (między każdą parą węzłów występuje dokładnie jedna droga). Sposób implementacji był analogiczny jak w sieci testowej - dodawaliśmy kolejne linijki kodu tworząc hosty, switchy i połączenia. Po stworzeniu struktury wyliczyliśmy opóźnienia na łączach wiedząc, że szybkość propagacji sygnału optycznego w światłowodzie wynosi około 200000 km/s i korzystając z wzoru:  $\frac{\sqrt{2} * \text{odleglosc}}{200km/ms}$ . Na poniższym rysunku znajduje się topologia naszej sieci wraz z wybranymi miastami i policzonymi opóźnieniami.



Rys. 1. Topologia sieci wykonana przy pomocy strony <https://www.smartdraw.com/>

Podobnie jak przy sieci testowej przenieśliśmy napisany kod na maszynę wirtualną, a następnie uruchomiliśmy naszą sieć w Mininet. Na koniec komendą `pingall` sprawdziliśmy połączenie między każdym z hostów.

```

mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)

```

Rys. 2. Sprawdzenie połączenia pomiędzy wszystkimi hostami

## 2. Zadanie 2

### 2.1. Zależność między opóźnieniem a RTT

W następnym zadaniu należało określić drogę, którą pakiet kontrolny przemierza pomiędzy hostami. Z racji drzewiastej struktury (możliwa tylko jedna droga między węzłami) skupiliśmy się na określeniu, czy czas RTT z jakim otrzymujemy odpowiedź na wysłanie pakietu kontrolnego wynosi w przybliżeniu dwukrotność opóźnień na łączach między węzłami. Oprócz opóźnień na łączach założyliśmy minimalne (równe 0,1 ms) opóźnienie na switchach. Pierwszą parą hostów, którą poddaliśmy eksperymentom była para h5-h8. Opóźnienie jakie napotyka pakiet kontrolny na odcinku h5-h8 wynosi:  $0,1 + 1,61 + 2,97 + 3,57 + 2,47 + 0,1 = 10,82$  ms. Wiedząc, że RTT to czas od wysłania pakietu do otrzymania odpowiedzi zakładamy, że minimalny czas RTT musi wynosić  $2 * 10,82 = 21,64$ . Sprawdzenia naszej tezy dokonaliśmy wywołując w terminalu komendę: h5 ping h8 -c 50 (flaga -c z argumentem 50 oznacza wysłanie 50 pakietów kontrolnych). Otrzymany przez nas wynik udowadnia poprawność naszej tezy - czas RTT w żadnej próbie nie był krótszy od teoretycznego, a wartość średnia była niewiele wyższa od zakładanej.

```

--- 10.0.0.8 ping statistics ---
50 packets transmitted, 50 received, 0% packet loss, time 49070ms
rtt min/avg/max/mdev = 22.148/24.029/26.390/1.052 ms

```

Rys. 3. Rezultat wykonania pingów pomiędzy hostem h5 i h8

W następnym teście sprawdziliśmy przypadek odwrotny do przedstawionego powyżej, tj. wysłaliśmy pakiety kontrolne z hosta h8 do hosta h5 oczekując otrzymania zbliżonego wyniku do tego z relacji h5-h8. Uzyskane dla obu sytuacji wyniki nieznacznie się różnią, co pokazuje że bez znaczenia w którą stronę podróżuje pakiet przy niezmiennych innych parametrach czas RTT nie zmienia się.

```

--- 10.0.0.5 ping statistics ---
50 packets transmitted, 50 received, 0% packet loss, time 49064ms
rtt min/avg/max/mdev = 22.223/23.754/26.594/0.981 ms

```

Rys. 4. Rezultat wykonania pingów pomiędzy hostem h8 i h5

Aby upewnić się co do ścisłej zależności występującej między wartością opóźnienia napotykanego na drodze pakietu a czasem RTT przeprowadziliśmy eksperyment, w którym zmieniliśmy delay pomiędzy hostami i switchami z 0,1 ms na 0,3 ms, nie zmieniając pozostałych wartości. Oczekiwaliśmy, że otrzymany czas RTT będzie większy o 0,8 ms. Jak możemy zauważyć otrzymany średni wynik wyniósł ok. 24,8 ms, co w porównaniu do sytuacji pierwotnej (ok. 24 ms) daje niemal dokładną różnicę wynoszącą 0,8 ms.

```

--- 10.0.0.8 ping statistics ---
50 packets transmitted, 50 received, 0% packet loss, time 49068ms
rtt min/avg/max/mdev = 23.083/24.801/26.844/1.007 ms

```

Rys. 5. Rezultat wykonania pingów pomiędzy hostem h5 i h8 z zwiększonym parametrem delay

Podobne testy wykonaliśmy dla innych par hostów - tym razem bez zmiany oryginalnych danych. Chcieliśmy tym samym pokazać, że uzyskany rezultat dla pary h5-h8 nie był przypadkiem, czego dowodzą uzyskane wyniki zamieszczone w poniższej tabeli:

hosty	teoretyczny RTT [ms]	RTT min. [ms]	RTT śr. [ms]	RTT max. [ms]	Opóźnienie pomiędzy h i s [ms]
h5-h8	21,62	22,148	24,029	26,39	0,1
h8-h5	21,62	22,223	23,754	26,594	0,1
h5-h8	22,42	23,083	24,801	26,844	0,3
h1-h2	6,34	6,497	7,344	15,181	0,1
h5-h9	25,54	26,694	28,535	34,122	0,1
h6-h7	20,52	21,297	23,13	25,459	0,1

Tabela 1. Pozostałe testy

## 2.2. Badanie zależności pomiędzy RTT a przepustowością

W tym eksperymencie skupimy się na parze hostów h1-h2. Na potrzebę tego testu zwiększyliśmy wielkość wysyłanego pakietu kontrolnego do maksymalnej wartości tj. 65 507 B ( skorzystaliśmy z opcji "-s 65507"), tak aby zaobserwować potencjalne przeciążenie sieci. Przy stałej wielkości wysyłanego pakietu zmniejszaliśmy przepustowość analizując przy tym zmiany RTT a przy okazji ewentualne straty pakietów. Pierwszy test wykonaliśmy dla oryginalnej przepustowości (100Mb/s). Uzyskany rezultat, który widać na rysunku poniżej, znacząco odbiega od czasu RTT wynikającego z sumowania opóźnień na łączach (6,34 ms). Nie wystąpiły też straty.

```
mininet> h1 ping -s 65507 h2 -c 50 -q
PING 10.0.0.2 (10.0.0.2) 65507(65535) bytes of data.

--- 10.0.0.2 ping statistics ---
50 packets transmitted, 50 received, 0% packet loss, time 49100ms
rtt min/avg/max/mdev = 17.023/18.053/33.523/2.849 ms
```

Rys. 6. Rezultat wykonania pingów pomiędzy hostami h1 i h2 przy wysłaniu dużego rozmiaru ping

W kolejnych zmianach wartości przepustowości łącza (zmniejszaliśmy je za każdym razem dziesięciokrotnie) obserwowaliśmy wzrost czasu RTT, ilość utraconych pakietów nie zmieniała się (wynosiła 0). Dopiero przy zmianie przepustowości na 0,1 Mb/s zaobserwowaliśmy stratę wynoszącą 76 % wszystkich wysyłanych pakietów. Ostatnim krokiem było sprawdzenie czy delikatne zwiększanie przepustowości sprawi, że utrata pakietów będzie mniejsza. Ustawiliśmy parametr bw na 0,3 i w rezultacie otrzymaliśmy stratę pakietów wynoszącą 22 %, co potwierdziło nasze przewidywania.

```
mininet> h1 ping -s 65507 h2 -c 50 -q
PING 10.0.0.2 (10.0.0.2) 65507(65535) bytes of data.

--- 10.0.0.2 ping statistics ---
50 packets transmitted, 12 received, +14 errors, 76% packet loss, time 49806ms
rtt min/avg/max/mdev = 10411.110/46144.718/106100.936/32161.475 ms, pipe 42
```

Rys. 7. Wykonanie ping przy ustawieniu przepustowości na 0,1 Mb/s

W poniższej tabeli znajdują się wszystkie testy wykonane dla hostów h1 i h2, polegające na wysyłaniu pingu wielkości 65507 bajtów dla różnych wartości przepustowości łączy. W tabeli znajduje się również wartość RTT (minimalna, średnia i maksymalna) oraz procent utraconych pakietów podczas transmisji.

opóźnienie [ms]	min [ms]	śr. [ms]	max [ms]	utracone pakiety [%]
100	17,023	18,053	33,523	0
10	110,657	112,749	123,24	0
1	1047,086	1048,781	1056,284	0
0.1	10411,11	46144,718	106100,936	76
0.5	2088,919	3849,265	5618,777	0
0,3	3719,192	20301,549	41180,864	22

Tabela 2. Rezultaty wykonanych testów

Zaobserwowaliśmy, że dla stałej wielkości przesyłanego pakietu, do momentu obserwowania strat, przy dziesięciokrotnym zmniejszeniu przepustowości, czas RTT rósł około dziesięciokrotnie.

### 2.3. Wnioski

Wykonane eksperymenty pozwoliły nam empirycznie sprawdzić zależności występujące pomiędzy parametrami przesyłu danych a charakterystykami łącz podczas przesyłania pakietów siecią. W pierwszej części tego zadania udowodniliśmy, że suma opóźnień występujących na łączach między hostami pomnożona przez dwa jest w przybliżeniu równa czasowi RTT. Co więcej, wyniki drugiej części eksperymentów utwierdziły nas w przekonaniu o ścisłym związku przepustowości z czasem RTT, a coraz większe zmniejszanie parametru bw doprowadziło w ostatecznym kroku do zaobserwowania strat pakietów - po osiągnięciu pewnej wartości krytycznej i dalszym zmniejszaniu przepustowości straty stawały się coraz większe. Eksperymenty mogłyby zostać przeprowadzone w bardziej realistyczny sposób np. poprzez wysłanie większego pakietu (ograniczenia komendy ping zmusiły nas do zmiany parametru przepustowości, zamiast zwiększania wielkości pakietu).

## 3. Zadanie 3

W ramach przygotowania do drugiej części laboratorium skonfigurowaliśmy program xterm, dzięki któremu uruchomiliśmy terminale dla hostów h1 i h2 (co widać na rysunku poniżej). Następnie z powodzeniem połączyliśmy się pomiędzy hostami h1 i h2 za pomocą usługi telnet oraz ssh.



Rys. 8. 2 oddzielne terminale dla hostów h1 i h2

Na hoście h1 postawiliśmy pythonowy serwer webowy, a następnie pobraliśmy dane z serwera na hoście h2, co przedstawiają poniższe rysunki.

```
root@mininet-vmt:/home/mininet# python3 -m http.server 80 &
[1] 2842
root@mininet-vmt:/home/mininet# Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.0.0.2 - - [20/Oct/2023 07:45:33] "GET / HTTP/1.1" 200 -
```

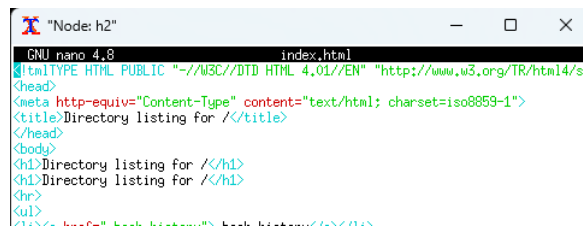
Rys. 9. Stawianie pythonowego serwera HTTP

```
root@mininet-vmt:/home/mininet# wget http://10.0.0.1:80
--2023-10-20 07:47:31-- http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1530 (1.5K) [text/html]
Saving to: 'index.html.1'

index.html.1      100%[=====] 1.49K  --.-KB/s  in 0s
2023-10-20 07:47:31 (7.77 MB/s) - 'index.html.1' saved [1530/1530]
```

Rys. 10. Pobieranie zawartości z serwera

W terminalu hosta h1 (na którym stał serwer), pojawiło się zapytanie GET, zaś na hoście h2 otrzymaliśmy informację kolejno o połączeniu się z serwerem, pobraniu jego zawartości i zapisaniu jej do pliku index.html, którego kawałek zamieściliśmy poniżej.



```
GNU nano 4.8 index.html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/s
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso8859-1">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<h1>Directory listing for /</h1>
<hr>
<ul>
</ul>
```

Rys. 11. Kawałek pliku index.html

Terminale działały poprawnie, wszystkie próby ich testowania, czyli łączenie się pomiędzy nimi, wysyłanie pinga oraz postawienie serwera i pobranie jego zawartości zakończyły się powodzeniem.

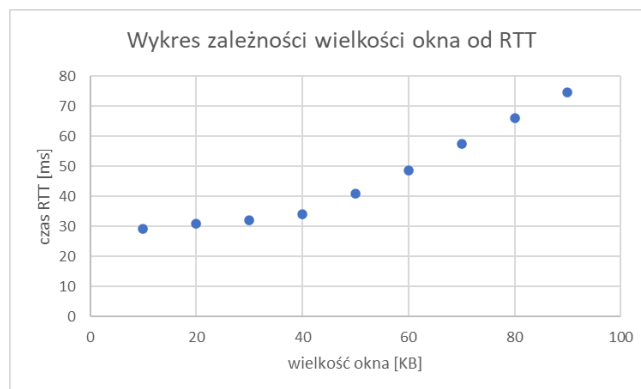
## 4. Zadanie 4

### 4.1. TCP

Eksperymenty oraz ich wyniki zostały opisane w niniejszym rozdziale. Analogiczne testy zostały przeprowadzone dla dwóch par hostów, a ostateczne wnioski zostały wyciągnięte z zależności występujących w obydwu relacjach.

#### 4.1.1. Zależność między wielkością okna a przepustowością i RTT

Pierwszym zbadanym przez nas parametrem była wielkość okna, za którą odpowiadała flaga -w. Przepustowości na wszystkich łączach zostały ustawione na 10 Mb/s, dla ułatwienia operowania resztą parametrów. W tym eksperymencie skupiliśmy się na relacji h5-h10 (Bruksela-Budapeszt), a naszym głównym zadaniem było sprawdzenie potencjalnej zależności występującej między wielkością okna a przepustowością z jaką wysyłane były dane oraz średnim czasem RTT. Nasze testy rozpoczęliśmy od ustawienia wielkości okna na 70 KB (zarówno na serwerze jak i kliencie) oraz wielkości wysyłanego pliku na 40 MB. Otrzymane wyniki (przepustowość: 9,55 Mb/s oraz RTT: 57,615 ms) pokazywały, że mimo praktycznie maksymalnego wykorzystania łącza czas RTT jest zdecydowanie większy od jego teoretycznej wartości (ok. 24,76 ms). Zmniejszając wielkość okna o 10 KB analizowaliśmy kolejne wyniki - dla następnych wartości (60 KB, 50 KB, 40 KB) nie zaobserwowaliśmy spadku przepustowości, natomiast za każdym razem znaczącemu zmniejszeniu ulegał czas RTT. Przy zmianie parametru na 30 KB mogliśmy dostrzec pierwszą widoczną zmianę przepustowości: 7,3 Mb/s. Kolejne próby dla coraz mniejszych wielkości okna dawały coraz mniejszy czas RTT, ale jednocześnie zdecydowanie wolniejszą przepustowość. Możemy tym samym wywnioskować, że istnieje zależność między wielkością okna a czasem RTT oraz przepustowością. Początkowy spadek RTT przy niezmienniej przepustowości wskazuje, że wielkość okna nadal była wtedy większa od optymalnego (według testów ok. 40 KB - ostatnia wartość, dla której średnia przepustowość osiągała powyżej 9 Mb/s), natomiast w momencie przekroczenia wartości optymalnej nasza transmisja zaczynała tracić na jakości - malejąca przepustowość wskazywała na niepełne wykorzystanie posiadanych zasobów łącza. Poniżej zamieszczamy wykres zależności wielkości okna od czasu RTT. Zauważamy, że dla okna większego od 40KB wykres przyjmuje zależność liniową, natomiast dla mniejszych stabilizuje się (nie może maleć w nieskończoność, gdyż zaprzeczyłoby to wcześniej udowodnionej zależności między opóźnieniem na łączach a czasem RTT).



Rys. 12. Wykres zależności wielkości okna od czasu RTT

#### 4.1.2. Zależność między wielkością bufora a przepustowością

Kolejnym etapem naszych eksperymentów było dodanie parametru `-l` (wielkości bufora). Dla standardowej wartości innych parametrów ustawiliśmy wpięrow wielkość bufora na 1 B (`-l 1`). Uzyskane rezultaty wykazywały, że przepustowość (3,87 Mb/s) była daleka od możliwej, wynikającej z charakterystyk łącza. Kolejno zwiększaliśmy wielkość bufora o 1 MB i dostawaliśmy następujące wyniki: 7,56 Mb/s, 11,2 Mb/s, 13,2 Mb/s. Dla kolejnych wielkości bufora średnie wartości przepustowości nie zmieniały się już tak dramatycznie. Z powyższego doświadczenia możemy zatem wywnioskować, że w początkowych wielkościach (1 i 2) bufor był niewystarczająco duży na pełne wykorzystanie możliwości łącza. Dopiero w momencie zwiększenia wielkości bufora przepustowość wzrosła, co skutkowało zwiększeniem jakości transmisji. Zaobserwowaliśmy również, że dla zwiększenia wielkości bufora do np. 15 B chwilowe zmiany przepustowości po stronie klienta stają się zdecydowanie bardziej prawdopodobne (średnia ich wartość to nadal ok. 14,3 Mb/s), natomiast przepustowość po stronie serwera nadal jest niezmienna - nie przekracza granicy 10Mb/s.

#### 4.1.3. Badanie zmian przepustowości danych po dodaniu flagi `-b`

Następnie skupiliśmy się na analizie parametrów ruchu, gdy dodany został parametr `-b` (bit rate). Zauważyliśmy, że przy ustawieniu parametru na wartość mniejszą od przepustowości łącza (ta nadal wynosiła 10 Mb/s) obserwowaliśmy ograniczenie przepustowości do wartości ustawionej za flagą `-b` (wykonaliśmy test 8 Mb/s i 3 Mb/s). Dla utwierdzenia się w przekonaniu, co do poprawnej interpretacji tego parametru zmieniliśmy wartości przepustowości dla wszystkich łączy na drodze pomiędzy h5-h10 na 5Mb/s. Tym samym, dla wartości bit rate `-b` ustawionej na 3 Mb/s ogranicza przepustowość (jak w poprzednim przypadku), natomiast gdy wartość ustawiona została na 8 Mb/s nie zaobserwowaliśmy ograniczenia przepustowości. Tym samym druga próba potwierdziła nasze przypuszczenia - mniejsza wartość bit rate od przepustowości łącza ogranicza przepustowość do zadanej wartości.

#### 4.1.4. Zmiana przepustowości na jednym z łączy

Zaciekawiło nas również jak zachowa się przepustowość transmisji, gdy zmianie ulegnie przepustowość na jednym z łączy, którymi podróżuje pakiet. W celu uzyskania odpowiedzi na postawiony wyżej problem zmniejszyliśmy przepustowość łącza między miastami Dortmund-Berlin (s2-s1) na 5 Mb/s. Po uruchomieniu skryptu z resztą parametrów ustawioną na domyślne zauważyliśmy średnią przepustowość po stronie klienta wynoszącą 6,67 Mb/s. Analizując logi z kolejnych sekund trwania sesji zauważyliśmy występujące ponowne przesłania utraconych pakietów - łącze było przeciążone i obserwowane były duże zmiany chwilowej przepustowości. W tym samym czasie przepustowość po stronie serwera wyniosła ok. 5 Mb/s. Wywnioskowaliśmy z tego, że ograniczenie przepustowości na jednym z łączy wpływa na całość transmisji i może doprowadzić do pogorszenia jakości transmisji danych.

Analogiczne testy wykonaliśmy dla pary hostów **h4-h6 (Gdańsk - Amsterdam)**, a uzyskane wyniki utwierdziły nas w przekonaniu, że rezultaty uzyskane w eksperymentach na relacji h5-h10 nie były przypadkiem.

#### 4.1.5. Wnioski

Eksperymenty zaproponowane przez nasz zespół dowodzą, że parametry sesji TCP takie jak wielkość okna, wielkość bufora, czy ograniczona przepustowość bezsprzecznie wpływają na jakość sesji TCP. Po pierwsze nie-

wystarczająco duża wielkość okna może ograniczyć przepustowość sesji, a odpowiednie manipulowanie jego wielkością może zwiększyć jakość przesyłu danych. Co więcej, zauważyliśmy że równie istotne jest ustawienie odpowiedniej wielkości bufora - zbyt mały bufor może spowalniać przysyłanie kolejnych pakietów. Najciekawszym testem okazał się mimo wszystko ostatni, w którym sztucznie zmieniliśmy przepustowość na jednym z łączy - ta jedna zmiana znacząco wpłynęła na spadek jakości całego przesyłu.

## 4.2. UDP

### 4.2.1. Badanie block\_rate na jakość transmisji

Analizę dla UDP rozpoczęliśmy od badania wpływu parametru block\_rate "-b" na przepustowość dla relacji serwera h1 i klienta h2 o ustawionej przepustowości 10 Mb/s. Podzieliliśmy ten test na 3 etapy, czyli kiedy parametr block rate jest mniejszy od optymalnej wielkości, równy jej oraz większy od niej. Dla wartości mniejszych, zgodnie z przewidywaniami, nie odnotowaliśmy żadnych strat. Zaobserwowaliśmy również wzrost parametru bandwidth wraz z zwiększaniem ilości pakietów wysyłanych na sekundę (pps). Jest to spodziewany efekt, ponieważ, zwiększając ilość pakietów możemy wysłać więcej danych, a to powoduje większe wykorzystanie dostępnej przepustowości. Dla ustawienia wartości block\_rate na granicy możliwości sieci, straty były niewielkie i wynosiły 3,8 %. Uzyskaliśmy przepustowość 9,41 Mb/s, co przekonało nas, że optymalna przepustowość powinna być delikatnie mniejsza od maksymalnej, żeby nie odnotowywać strat. Następnie ustawiliśmy parametr block rate na 800 pps, w efekcie czego odnotowaliśmy straty na poziomie 9,8 %. Dla block\_rate ustawionego na 900 pps straty wynosiły już 12 %, dla parametru block\_rate ustawionego na 1000 pps straty osiągnęły 27%, a po zmianie parametru na wartość 1500 pps straty doszły do 50%. Dla każdego testu, w którym ustawiliśmy parametr block\_rate na większą wartość od 720, optymalna przepustowość wynosiła około 9,4 Mb/s. Po wykonaniu tych testów możemy również stwierdzić, że parametr bandwidth wzrasta wprost proporcjonalnie do wzrostu ilości wysyłanych pakietów aż do maksymalnego wykorzystania przepustowości sieci.

### 4.2.2. Badanie wpływu wielkości bloku na jakość transmisji

W kolejnym teście badaliśmy wpływ parametru -l odpowiadającego za rozmiar wysyłanych bloków danych na jakość transmisji (QoS). Zaczęliśmy od ustawienia wartości -l na 50. Odnotowaliśmy straty na poziomie 82 % i przepustowość 1,79Mb/s. Następnie ustawiliśmy parametr -l na 100, co skutkowało obniżeniem się strat do 71 % oraz podwyższeniem przepustowości do 3Mb/s. Postanowiliśmy dalej zwiększać parametr -l, więc ustawiliśmy go na 200. Przepustowość wyniosła 4,78 Mb/s a straty 54 %. Postanowiliśmy zwiększać parametr -l do momentu osiągnięcia zerowych strat. Prawie się to udało dla wartości -l ustawionej na 1300 (straty były wtedy bliskie 0). Przepustowość łącza wyniosła wtedy 9,41 Mb/s. Następnie zwiększyliśmy -l na 1500, w efekcie czego straty zaczęły rosnąć do 7,7 %, zaś przepustowość łącza pozostawała na podobnym poziomie. Z tych testów wywnioskowaliśmy, że przy przysyłaniu danych protokołem UDP, ważne jest, żeby dobrze dobrać wielkość obsługiwanych bloków. Ustawienie za małej wartości skutkuje przepełnieniem się bufora i pojawieniem się strat oraz spadkiem przepustowości. Ustawienie za dużej wartości również nie jest pożądane, ponieważ proces fragmentacji IP wymaga dodatkowego czasu, co skutkuje obniżeniem się jakości świadczonej usługi.

### 4.2.3. Zmiana przepustowości na jednym z łączy

Postanowiliśmy jeszcze sprawdzić co się stanie, jeżeli zmniejszymy przepustowość na jednym z łączy, przez które będzie przechodził transfer danych. Wybraliśmy relację h1 (serwer) h9 (klient). Na łączu pomiędzy s3 i s8 zmniejszyliśmy parametr bw o połowę i wyniósł on 5 Mb/s. Odnotowaliśmy straty na łączu rzędu 40 % oraz spadek przepustowości łącza do poziomu 4,83 Mb/s. Wywnioskowaliśmy z tego, że przesył danych musi iść jedną drogą (potwierdzenie tego, że struktura sieci jest drzewiasta) oraz, że spadek przepustowości na jednym łączu pogorszy jakość transmisji dla wszystkich relacji przechodzących przez te łącze.

Wszystkie powyżej opisane testy wykonaliśmy również dla hostów **h3-h9**. Wyniki, które uzyskaliśmy pozwoliły wysnuć te same wnioski na temat wpływu ustawień sieci na jakość transmisji.

### 4.2.4. Wnioski

Wykonane przez nas testy utwierdziły nas w przekonaniu, że aby sieć działała wydajnie dla UDP, należy odpowiednio dobrać takie parametry jak ilość pakietów wysyłanych w ciągu sekundy, czy rozmiar bloku danych. Mają one bezpośredni wpływ na QoS i dobranie ich w nieprawidłowy sposób obniża jakość transmisji. Sieć staje się wolniejsza oraz pojawiają się straty.



## 5. Zadanie 5

### 5.1. TCP vs TCP

#### 5.1.1. Badanie jakości połączenia przy kilku sesjach TCP przechodzących przez to samo łącze

W pierwszym teście postanowiliśmy puścić jednocześnie 2 połączenia TCP, które mają kawałek wspólnej trasy. W tym teście klientami były hosty h10 i h9, a serwerem był host h1. Początkowo każdy z klientów próbował wykorzystać maksymalnie możliwości łącza, jednak szybko doszło do spadku przepustowości łącza i obaj klienci dzielili się przepustowością sprawiedliwie (sieć miała przepustowość 10Mb/s, a każdy z klientów wysyłał z prędkością 4,68 Mb/s). Po stronie serwera widzieliśmy również sprawiedliwy podział przepustowości - każda z sesji otrzymała średnio ok. 4.5 Mbit/s. Aby potwierdzić występowanie równego podziału przepustowości zmniejszyliśmy przepustowość na wszystkich łączach na 5 Mb/s. Tym samym wyniki spełniały zasadę +/- równego podziału przepustowości - zaobserwowaliśmy dwukrotny spadek średniej wartości przepustowości dla obu sesji.

Następnie do skryptu z poprzedniego testu dodaliśmy jeszcze 2 klientów, którymi były hosty h7 i h8. Podobnie jak w poprzednim teście, puściliśmy jednocześnie połączenie dla 4 hostów i początkowo każdy z hostów chciał w pełni wykorzystać przepustowość, jednak zaledwie moment od rozpoczęcia testu hosty podzieliły się przepustowością łącza kolejno 2,2 Mb/s, 2,15 Mb/s, 2,22 Mb/s, 2,21 Mb/s. Pojawiły się tutaj drobne różnice w przepustowości jednak tak jak z poprzedniego testu, wnioskujemy, że dostępna przepustowość została podzielona sprawiedliwie.

#### 5.1.2. Podział przepustowości w TCP dla ustawionych różnych wielkości okien

W kolejnym teście wróciliśmy do 2 hostów, h9 i h10. Ustawiliśmy dla nich wielkości okna kolejno 30KB oraz 5KB. Host z większą wielkością okna uzyskał przepustowość 8,3 Mb/s, zaś z mniejszą wielkością okna 1,14 Mb/s. Warto tutaj odnotować, że sesje nie trwały tyle samo czasu i sesja z mniejszym oknem skończyła się później przez co w ostatniej fazie (gdy pracowała już sama) nastąpił nieznaczny wzrost przepustowości (z 800 Kbit/s na 1,2 Mbit/s). Zauważamy tym samym, że w przypadku różnych wielkości okna podział nie jest równy, na czym korzysta sesja o większej wielkości okna.

#### 5.1.3. Ograniczenie przepustowości na jednym łączu

Ostatni test przeprowadziliśmy wracając do wyjściowych wartości przepustowości na wszystkich łączach, zmieniając jedynie przepustowość na łączu s1-s3 na 5 Mbit/s, mogącą powodować utrudnienia w transmisji. Konfiguracja relacji pozostała ta sama: serwer - h1 i klienci - h9 i h10. Uzyskany rezultat wskazuje, że przepustowość na serwerze wynosiła kolejno: 2,36 Mbit/s oraz 2,58 Mbit/s. Przepustowość po stronie klientów była chaotyczna - początkowe próby wysyłania z większą przepustowością zostały uniemożliwione po przeciążeniu łącza s3-s1.

Analogiczne testy wykonaliśmy dla hostów **h2-h7-h8 (Dortmund - Praga - Wiedeń)**, a uzyskane wyniki utwierdziły nas w przekonaniu, że rezultaty uzyskane w eksperymentach na relacji h1-h9-h10 nie były przypadkiem.

#### 5.1.4. Wnioski

Jak możemy zauważyć, gdy dwie lub więcej sesje TCP korzystają z tego samego łącza zachodzi równy podział przepustowości łącza (dla  $k$  transmisji przepustowość jednej z nich wynosi:  $\frac{bandwidth}{k}$ ). Wyjątkiem okazała się sytuacja, gdy po ustawieniu różnych wielkości okna pierwszeństwo miała ta z większą wielkością. Zgodnie z przewidywaniami, występowanie większej liczby transmisji ma wpływ na ich jakość i potencjalne przeciążenie sieci.

### 5.2. UDP vs UDP

#### 5.2.1. Badanie jakości połączenia przy kilku sesjach UDP przechodzących przez to samo łącze

W pierwszym teście puściliśmy w tym samym czasie 2 połączenia, które częściowo szły przez tą samą trasę. Były to pary hostów h5-h1 oraz h6-h1 w relacji klient serwer. Przepustowość sieci była ustawiona na 10 Mb/s i ustawiając wartości sumarycznie mniejsze dla poszczególnych połączeń, nie odnotowaliśmy żadnych strat (testowaliśmy sieć przy ustawieniu zarówno tych samych jak i różnych parametrów -b dla poszczególnych relacji na

przykład 6Mb/s pomiędzy h5 i h1 oraz 3 Mb/s pomiędzy h6 i h1). W kolejnym kroku ustawiliśmy parametry -b w obu połączeniach na te same wartości, które sumarycznie odpowiadały przepustowości sieci. Odnotowaliśmy pierwsze straty oraz spadek przepustowości. Zauważyliśmy, że dostępna przepustowość nie została podzielona po równo. W relacji h5-h1 wyniosła ona 4,12 Mb/s oraz pojawiło się 19 % strat, zaś w relacji h6-h1 przepustowość wyniosła 4,42 Mb/s, a straty wyniosły 13 %. Test powtórzyliśmy i uzyskaliśmy nieznacznie różniące się wyniki. Warto dodać, że klienci próbowali przez cały okres trwania sesji wysyłać dane z prawie tą samą przepustowością, a nie tak jak w TCP zmniejszyć ją, żeby nie odnotowywać strat. Następnie zwiększyliśmy parametr -b do wartości 10 Mb/s na obu połączeniach. Klienci ponownie w trakcie trwania sesji wysyłali dane z tą samą przepustowością, zaś po stronie serwera odnotowaliśmy straty (58% dla h5-h1 oraz 55% dla h6-h1) oraz zmniejszoną przepustowość, wynoszącą 4,18 Mb/s dla h5-h1 oraz 4,47 Mb/s dla h6-h1.

W kolejnym teście ustawiliśmy parametr -b dla h5-h1 na 5 Mb/s, a dla relacji h6-h1 na 15 Mb/s. Obaj klienci wysyłali cały czas dane z ustawioną dla nich przepustowością. Serwer zaś odbierał od h5 z prędkością 3,08 Mb/s (starty wyniosły 39 %), a od h6 z prędkością 6,6 Mb/s (starty wyniosły 56 %). Mimo 3-krotnej różnicy pomiędzy początkowymi wartościami bw, szybkość z jaką odbierał serwer różniła się około 2-krotnie.

Testy powtórzyliśmy dla 4 sesji jednocześnie. Uruchomiliśmy h7, h8, h9, h10 jako klientów oraz h1 jako serwer. Podobnie jak w poprzednich testach, dopóki nie przekraczaliśmy dostępnej przepustowości, to każde z połączeń nie miało strat i wykorzystywało ustawioną swoją przepustowość w pełni. Przy przekroczeniu dostępnej przepustowości pojawiły się straty różne dla poszczególnych hostów (z przedziału 19% do 43%) oraz inne parametry bandwidth z zakresu od 2,27 Mb/s do 3,29 Mb/s.

### 5.2.2. Zmniejszenie przepustowości na łączy, przez które przechodzi transmisja

W tym teście zmniejszyliśmy przepustowość na łączy pomiędzy S1 i S3 z parametru bw ustawionego na 10 na 5. Puściliśmy jednocześnie z 2 hostów, h7 i h9 transmisje do hosta h1. Odnotowaliśmy obniżenie przepustowości po stronie serwera, do takiej wartości, która nie przekracza nowo ustawionej wartości bw. Klienci zaś wysyłali swoje dane z tą samą prędkością nie zważając na straty. Eksperyment ten dowiódł, że transmisja danych musi iść jedną, wyznaczoną drogą, co potwierdza, że nasza sieć ma strukturę drzewiastą.

### 5.2.3. Wnioski

Z powyższych testów wywnioskowaliśmy, że dla UDP, w momencie przeciążenia łączy, przepustowość nie jest dzielona uczciwie tak jak miało to miejsce w przypadku TCP. Te same testy wykonaliśmy dla relacji h7-h1 oraz h9-h1 puszczając jednocześnie transmisje. Otrzymane rezultaty, pozwalają wydedukować te same wnioski.

## 5.3. UDP vs TCP

Wykonaliśmy test uruchamiając jednocześnie połączenie TCP oraz UDP zmieniając parametr block size dla TCP. Przy ustawionej małej wartości block size, zaobserwowaliśmy, że TCP do momentu zakończenia sesji UDP, miała małą przepustowość rzędu 200 KB, zaś po zakończeniu sesji UDP podskoczyło do około 1 MB. Po zwiększeniu parametru block size zaobserwowaliśmy znacznie zwiększony bandwidth (do ponad 2 Mb/s) dla TCP, który i tak znacznie wzrósł po zakończeniu sesji UDP. W TCP również była mniejsza potrzeba ponownego wysyłania danych, po zwiększeniu block size. Podobny efekt przewidywaliśmy, ponieważ zwiększając rozmiar bloków w TCP, wysyłamy więcej danych i bufor się szybciej napełnia, przez co sesja w TCP znacząco zwalnia. Przy ustawieniu parametru block size na większe wartości, bufor nie napełni się tak szybko i nie odnotujemy strat, które w przypadku TCP trzeba niwelować (ponownie wysłać dane). Istotne jest również, że połączenie UDP nie zwalniało przy pojawieniu się strat, tylko wysyłało dane dalej.

## 6. Podsumowanie

Wykonanie ćwiczenia laboratoryjnego pozwoliło nam lepiej zrozumieć różnice między protokołami TCP oraz UDP. Dobieranie odpowiednich parametrów ustawiając charakterystykę sieci zagłębiło nas w bardziej szczegółowe poznanie takich zagadnień jak wielkość bloku, rozmiar okna, czas RTT. Są to niewątpliwie ważne aspekty pracy z siecią, które są obowiązkową wiedzą w cyberbezpieczeństwie.