Sieci i chmury teleinformatyczne

Laboratorium nr 2 - Płaszczyzna sterowania sieci

Rafał Dadura, Juliusz Kuzyka

Listopad 2023

Spis treści

1 Cel laboratorium

Głównym celem tego laboratorium jest zdobycie wiedzy na temat możliwości zarządzania siecią za pomocą kontrolera ONOS oraz umiejętności tworzenia aplikacji do sterowania siecią, które pozwalają na wpływanie na parametry przesyłanego w sieci ruchu.

2 Zadanie 1

Na początku przygotowaliśmy plan infrastruktury łączącej 10 miast europejskich, któyą opracowaliśmy w poprzednim laboratorium, jednak teraz mieliśmy nie ograniczać się już do grafu sieci będącego drzewem, więc zmieniliśmy go w następujący sposób.

Rysunek 1: Plan infrastruktury (niebieskie linie - połączenia, które dodaliśmy)

Następnie dodaliśmy te poołączenia do skryptu, którego używaliśmy w pierwszym laboratorium. Kolejnym zadaniem było skonfigurowanie węzłów w pełnej sieci z poprzedniego ćwiczenia, wykorzystując kontroler ONOS przy użyciu REST, abyśmy mogli przeprowadzić odpowiednie testy jak w poprzednim laboratorium. Wszystkie switche zostały przez nas skonfigurowane, a poniżej zamieściliśmy część przykładowego skryptu json, który odpowiadał za konfigurację switcha 1.

```
{
  "flows": [
    {
      "priority": 40000,
      "timeout": 1500,
      "isPermanent": true,
      "deviceId": "of:0000000000000001",
      "treatment": {
        "instructions": [
          {
            "type": "OUTPUT",
            "port": "1"
          }
        ],
        "deferred": []
```

```
      },
      "selector": {
        "criteria": [
          {
            "type": "IN_PORT",
            "port": "4"
          },
          {
            "type": "ETH_TYPE",
            "ethType": "0x0800"
          },
          {
            "type": "IPV4_DST",
            "ip": "10.0.0.1/32"
          },
          {
            "type": "IPV4_SRC",
            "ip": "10.0.0.10/32"
          }
        ]
      }
    },
   ...
}
```

Następnie zgodnie z poleceniem zdefiniowaliśmy i uruchomiliśmy skrypt *.bat z sekwencjami wywołań narzędzia curl konfigurującymi węzły sieć w połączeniu h1-h10, który wyglądał następująco:

```
curl --user karaf:karaf -X POST "http://192.168.0.191:8181/onos/v1/flows?appId=0000000000000001"
-d @switch11.json -H "Content-Type: application/json" -H "Accept: application/json"
curl --user karaf:karaf -X POST "http://192.168.0.191:8181/onos/v1/flows?appId=0000000000000001"
-d @switch12.json -H "Content-Type: application/json" -H "Accept: application/json"

pause
```

Korzystając z wcześniej opracowanych skryptów emulatora Mininet uruchamiających narzędzie iperf, rozpoczęliśmy realizację ponownych testów, aby potwierdzić możliwość realizacji sesji transportowych zdefiniowanych w ramach punktu 4 poprzedniego laboratorium oraz porównać wartości uzyskiwanych parametrów przesyłu danych. Stworzyliśmy kilka plików bat, które opowiadały ścieżkom badanym przez nas.

2.1 UDP

2.1.1 Parametr block_rate

Jako pierwszy parametr do analizy wybraliśmy opcję -b, która służy do określenia ilości pakietów na sekundę (pps). Poniżej zamieszczamy tabelę przedstawiającą wyniki testów z wybranymi wartościami tego parametru, kiedy przyjmował on wartości odpowiednio: 1500, 1200 i 700 w kontekście komunikacji między hostem h6 a h9.

badanie wpływu parametru -b

	Transfer (Mbytes)	bandwith (Mbits/s)	jitter (ms)	lost (%)	Latency avg (ms)	pps	NetPwr
większy niż przepustowość (-b 1500)	22,1	8,56	0,447	23	1601,837	1070	0,67
równy przepustwości (-b 1200)	21,6	7,68	3,149	5,8	1160,711	960	0,83
mniejszy niż przepustowość (-b 700)	13,2	5,54	0,232	1	12,162	692	56,96

Rysunek 2: Tabela statystyk parametru -b w UDP

Podczas naszych badań zauważyliśmy różne zachowania w zależności od parametru *block rate*, który odnosi się do ilości pakietów na sekundę (pps). Oto nasze obserwacje:

- **Gdy parametr *block rate* był mniejszy od wartości domyślnej** - zauważyliśmy brak strat w przepustowości, co było zgodne z naszymi oczekiwaniami.
- **Gdy parametr *block rate* był ustawiony na około wartość domyślną** - na początku zaobserwowaliśmy niewielkie straty wynoszące około 5,8%. Przy tym ustawieniu przepustowość wyniosła 7,68 Mb/s. Dla minimalizacji strat doszliśmy do wniosku, że ustawiona przez nas przepustowość powinna być mniejsza od wartości domyślnej.
- **Gdy parametr *block rate* przekraczał wartość domyślną** - zauważyliśmy, że wraz ze wzrostem wartości -b następował wzrost strat pakietów. Dla wartości 1500 straty wyniosły 23%.

Podsumowując, nasze badania wykazały, tak jak w poprzednim laboratorium, że parametr *block rate* ma wpływ na przepustowość oraz straty w sieci, a odpowiednie dostosowanie tego parametru może pomóc w zoptymalizowaniu wydajności transmisji danych.

2.1.2 Parametr wielkości bloku

Kolejnym parametrem, który poddaliśmy badaniom, był parametr "l,"służący do określenia wielkości bloku danych. Poniżej przedstawiamy tabelę podsumowującą wyniki testów dla parametru "l,"przy różnych ustawieniach: 1000, 500, 100 i 20, w kontekście komunikacji między h6 a h9.

badanie wpływu parametru -l

	Transfer (Mbytes)	bandwith (Mbits/s)	jitter (ms)	lost (%)	Latency avg (ms)	pps	NetPwr
-l 1000	21,6	7,68	3,149	5,8	1160,711	960	0,83
-l 500	11,3	4,76	0,875	0,87	9,72	1188	61,16
-l 100	2,27	0,952	0,327	0,89	5,129	1189	23,19
-l 20	0,465	0,19	0,268	0,8	4,017	1190	5,93

Rysunek 3: Tabela statystyk parametru -l w UDP

Wyniki badań, które przeprowadziliśmy były bardzo zbliżone do tych, które wyknaliśmy w ramach poprzedniego laboratorium. Również wywnioskowaliśmy, że większy rozmiar bloku danych może przyczynić się do zwiększenia przepustowości. Dzieje się tak, ponieważ przesyłanie mniej pakietów oznacza mniej nadmiarowych nagłówków pakietów, co przekłada się na bardziej efektywne wykorzystanie dostępnego pasma. Po drugie, zbyt duży rozmiar bloku może prowadzić do problemów związanych z fragmentacją, szczególnie w sieciach, gdzie istnieją ograniczenia dotyczące maksymalnego rozmiaru pakietu, co również prowadzi do większej utraty pakietów. Jak to widać w tabeli, dla -l 1000 straty wynosiły już ponad 5,8 %, gdzie dla mniejszych "l"nie przekraczały 1%.

2.2 TCP

Rozpoczęliśmy powtórkę naszych badań od analizy parametru -b, który określa prędkość przesyłu danych w trakcie testu prędkości transmisji TCP. Poniżej prezentujemy tabelę podsumowującą wyniki testów dla parametru -b, wraz z wybranymi wartościami: 10M, 5M, 4M, 1M i 0,5M, w kontekście komunikacji między h4 i h2.

2.2.1 Parametr max_bit_rate

badanie wpływu parametru -b

	Transfer (Mbytes)	bandwith (Mbits/s)	Reads	czas trwania (s)
-b 10M	10	3,36	5405	24,994
-b 5M	10	3,34	5442	25,142
-b 4M	10	3,33	5413	25,227
-b 1M	10	1,04	14190	80,727
-b 0,5M	10	0,52	20090	160,31

Rysunek 4: Tabela statystyk parametru -b w TCP

Parametr -b wpływa głównie na kontrolę prędkości przesyłu danych między klientem a serwerem w trakcie testu prędkości transmisji TCP. Gdy wartość parametru -b jest mniejsza od domyślnej, przepustowość maleje. Jak widać dla mniejszych parametrów -b przepustowości był znacznie mniejsze od tych, które miały ten parametr większy. Tyczy się to również czasu przesyłu danych. Również, gdy przyjmiemy wartości parametru -b powyżej 10M, przepustowość nie wzrasta, ponieważ napotyka ona ograniczenia narzucone przez sieć. Takie same wyniki, uzyskaliśmy w ramach pierwszego laboratorium.

2.2.2 Parametr volume

Kolejnym parametrem, który poddaliśmy testom, jest -n, służący do określenia ilości danych, które mają być przesłane podczas testu prędkości transmisji TCP. Poniżej przedstawiamy tabelę podsumowującą wyniki testów dla parametru -n, z wybranymi wartościami: 50M, 10M, 5M, 1M i 0,5M, w kontekście komunikacji między h4 i h2.

badanie wpływu parametru -n

	Transfer (Mbytes)	bandwith (Mbits/s)	Reads	czas trwania (s)
-n 50M	50	3,3	27271	127,041
-n 10M	10	3,36	5405	24,994
-n 5M	5	3,22	2708	13,033
-n 1M	1	3,64	551	2,305
-n 0,5M	0,5	3,61	275	1,163

Rysunek 5: Tabela statystyk parametru -n w TCP

Podobnie, jak w poprzednim laboratorium modyfikacja wartości parametru -n wpływa bezpośrednio na ilość danych, które zostaną przesłane. Im większa jest wartość parametru -n, tym większa ilość danych zostanie przesłana, co skutkuje dłuższym czasem trwania testu. Dla przykładu, czas trwania przesyłu danych dla -n 50M wynosił ponad 127 sekund, a dla -n 1M lekko ponad 2 sekundy.

2.2.3 Parametr block_size

Kolejnym parametrem, który poddaliśmy analizie, jest -l, służący do określenia rozmiaru bloków danych (lub segmentów), używanych podczas testu prędkości transmisji TCP. Poniżej przedstawiamy tabelę podsumowującą wyniki testów dla parametru -l, z wybranymi wartościami: 1, 2, 3, 4 i 5, w kontekście komunikacji między h4 i h2.

badanie wpływu parametru -l

	Transfer (Mbytes)	bandwith (Mbits/s)	Reads	czas trwania (s)
-l 5	10	8,48	3476	9,888
-l 4	10	8,6	3388	9,759
-l 3	10	7,86	3492	10,674
-l 2	10	6,66	3862	12,593
-l 1	10	3,36	5405	24,994

Rysunek 6: Tabela statystyk parametru -l w TCP

Rozmiar bloków danych ma istotny wpływ na wydajność transmisji TCP. Większe bloki danych mogą przyczynić się do zwiększenia przepustowości, ponieważ przesyłanie mniej pakietów oznacza mniej narzutu związanego z nagłówkami podczas transmisji, np. dla -l 5 przepustowość wynosiła 8,48 Mb/s, gdy dla -l 1 wynosiłą ona zaledwie 3,36 Mb/s. Dodatkowo, większe bloki pozwalają skrócić czas trwania transmisji, co jest zauważalne na przykładzie różnicy między -l 1 a -l 2.

Warto zaznaczyć, że powyżej wartości -l 5 zarówno czas trwania transmisji, jak i przepustowość nie zmieniały się znacząco. To sugeruje, że istnieje pewne optymalne wartości rozmiaru bloków danych, które zapewniają najlepszą wydajność transmisji. Do identycznych wniosków doszliśmy w ramach 1 laboratorium.

2.2.4 Parametr window

Kolejnym badanym parametrem był -w, który określa rozmiar bufora przesyłu danych wykorzystywanego podczas testu prędkości transmisji TCP. Poniżej prezentujemy tabelę podsumowującą wyniki testów dla parametru -w, z

wybranymi wartościami: 100000, 50000, 25000, 10000 i 5000, w kontekście komunikacji między h4 i h2.

badanie wpływu parametru -w

	Transfer (Mbytes)	bandwith (Mbits/s)	Reads	czas trwania (s)
-w 100000	10	3,27	5577	25,625
-w 50000	10	3,18	5358	26,361
-w 25000	10	3,2	5320	26,195
-w 10000	10	2,51	6445	33,368
-w 5000	10	0,582	22100	144,169

Rysunek 7: Tabela statystyk parametru -w w TCP

Rozmiar bufora przesyłu danych, określany za pomocą parametru -w, tak jak w laboratorium pierwszym, ma istotny wpływ na wydajność przepustowości w transmisji TCP. Zwiększanie rozmiaru bufora może znacząco poprawić przepustowość, na przykład dla -w 50000 przepustowość wynosiła 3,18 Mb/s, a dla 5000

Dodatkowo, rozmiar bufora ma wpływ na kontrolę narzutu związanego z protokołem TCP. Większy bufor może pomóc w zmniejszeniu narzutu związanego z nagłówkami TCP, co w konsekwencji przekłada się na poprawę przepustowości oraz skrócenie czasu trwania transmisji. Dlatego optymalny dobór rozmiaru bufora jest istotnym elementem optymalizacji wydajności transmisji TCP.

2.3 Wnioski

Warto zauważyć, że zarówno w przypadku protokołu TCP, jak i UDP, uzyskaliśmy rezultaty praktycznie identyczne jak te, które otrzymaliśmy w ramach ćwiczenia 1. To potwierdza poprawną konfigurację ścieżek przy użyciu styku REST oraz fakt, że w obu ćwiczeniach dane były przesyłane w dokładnie taki sam sposób. Dla poprawności wyników i porównań między różnymi eksperymentami kluczowe jest zachowanie spójności w konfiguracjach i procesach transmisji danych.

3 Zadanie 2

3.1 UDP

W poprzednim laboratorium w zadaniu 5, gdzie eksperymentowaliśmy z protokołem UDP, prowadziliśmy testy, tworząc równoległe połączenia między tymi samymi hostami. Tym razem wybraliśmy podobne połączenie do wcześniej rozważanego, czyli między h9 a h8. W nowej topologii ścieżki nie różnią się znacząco od poprzedniej. W pierwszym połączeniu droga to **h3-s3-s6-s8-h8**, a w drugim **h9-s9-s6-s8-h8**. Następnie powtórzyliśmy eksperymenty, korzystając z kontrolera Mininet, co pozwoliło nam na uzyskanie poniższych wyników dla pojedynczych połączeń (każde o przepustowości łącza 10 Mb/s):

- Dla h3-h8 - **8,40 Mb/s**, nie zaobserwowaliśmy strat pakietów.
- Dla h9-h8 - **8,67 Mb/s**, nie zaobserwowaliśmy strat pakietów.

Następnie odapliliśmy oba połączenia równolegle. Z racji, że połączenia nie dzieliły ze sobą ścieżek, przepustowość ich łącza wzrosła dwukrtonie. Przepustowości jakie otrzymaliśmy były:

- Parametr block_rate:
  - Dla tych samych parametrów nie przekraczających przepustowości sieci (2 połączenia równległe) - **4,18 Mb/s** (h3-h8 -b 4M),**4,19 Mb/s** (h9-h8 -b 4M), nie zaobserwowaliśmy strat pakietów.
  - Dla tych samych parametrów przekraczających przepustowość sieci (2 połączenia równległe) - **3,99Mb/s** (h3-h8 -b 8M),**4,13 Mb/s** (h9-h8 -b 8M), w obu przypadakach zaobserwowaliśmy straty na poziomie około 50%.
  - Różne parametry przepustowości nie przekraczające przepustowości sieci (4 połączenia równległe) - **2,09 Mb/s** (h3-h8 -b 2M), **2.09 Mb/s** (h3-h8 -b 2M), **2,09 Mb/s** (h9-h8 -b 2M), **2,03 Mb/s** (h9-h8 -b 2M), nie zaobserwowaliśmy strat pakietów.
  - Różne parametry przepustowości przekraczające przepustowości sieci (4 połączenia równległe) - **2,77 Mb/s** (h3-h8 -b 10M), **1,98 Mb/s** (h3-h8 -b 5M), **2,57 Mb/s** (h9-h8 -b 10M), **2,08 Mb/s** (h9-h8 -b 5M), połączenia z przepustowością 10 Mb/s miały straty pakietów na poziomie 70%, a te z parametrem -b 5M około 56%.

- Parametr block_size:
  - Różne parametry (4 połączenia równległe) - **2,03 Mb/s** (h3-h8 -l 5000), **0,56 Mb/s** (h3-h8 -l 100), **1,89 Mb/s** (h9-h8 -l 5000), **0,57 Mb/s** (h9-h8 -l 100), każde połączenie miało parametr -b 2M.

Potem przystąpiliśmy do określenia optymalnych tras w celu usprawnienia połączenia przy jednoczesnym wykorzystaniu nowej topologii. Nowymi trasami są:

- **h3-h8** - h3-s3-s8-h8
- **h9-h8** - h9-s9-s8-h8

Nasze testy ponownie rozpoczęliśmy od badania przepustowości podczas pojedynczego połączenia (każde o przepustowości łącza 10 Mb/s):

- Dla h3-h8 - **8,49 Mb/s**, nie zaobserwowaliśmy strat pakietów.
- Dla h9-h8 - **8,42 Mb/s**, nie zaobserwowaliśmy strat pakietów.

Następnie właczyliśmy połączenia równolegle. Z racji, że połączenia nie dzieliły ze sobą ścieżek, przepustowość ich łącza wzrosła dwukrtonie. Przepustowości jakie otrzymaliśmy były następujące:

- Parametr block_rate:
  - Dla tych samych parametrów nie przekraczających przepustowości sieci (2 połączenia równległe) - **8,75Mb/s** (h3-h8 -b 16M),**8,75 Mb/s** (h9-h8 -b 16M), oba połączenia miały starty pakietów na poziomie 38%.
  - Dla tych samych parametrów przekraczających przepustowość sieci (2 połączenia równległe) - **8,38Mb/s** (h3-h8 -b 8M),**8,38 Mb/s** (h9-h8 -b 8M), nie zaobserwowaliśmy strat pakietów.
  - Różne parametry przepustowości nie przekraczające przepustowości sieci (4 połączenia równległe) - **2,10 Mb/s** (h3-h8 -b 2M), **2,10 Mb/s** (h3-h8 -b 2M), **2,10 Mb/s** (h9-h8 -b 2M), **1,94 Mb/s** (h9-h8 -b 2M), nie zaobserwowaliśmy strat pakietów.
  - Różne parametry przepustowości przekraczające przepustowości sieci (4 połączenia równległe) - **5,64 Mb/s** (h3-h8 -b 10M), **2,88Mb/s** (h3-h8 -b 5M), **5,39 Mb/s** (h9-h8 -b 10M), **2,61 Mb/s** (h9-h8 -b 5M), wszystkie połączenia miały straty pakietów na poziomie 30%.
- Parametr block_size:
  - Różne parametry (4 połączenia równległe) - **4,06 Mb/s** (h3-h8 -l 5000), **1,20 Mb/s** (h3-h8 -l 100), **4,07 Mb/s** (h9-h8 -l 5000), **1,18 Mb/s** (h9-h8 -l 100), każde połączenie miało parametr -b 4M.

Wszystkie doświadczenia powtórzyliśmy ponownie dla połączenia h5-h3. Oryginalna droga jaką musiało pokonać te połączenie to **h5-s5-s4-s2-s1-s3-s6-s9-h9** oraz podobna droga, którą musieliśmy wybrać z takich samych, jak w przykładzie powyżej względów, **h7-s7-s4-s2-s1-s3-s6-h9**, zaś optymalne drogi jakie wybraliśmy to były: dla pierwszej drogi **h5-s4-s6-s9-h9**, a dla drugiej **h7-s4-s3-s8-s9-h9** . Powtarzając te eksperymenty doszliśmy jednak do takich samych wniosków.

3.1.1 Wnioski dla UDP

Wywnioskowaliśmy, że ustalanie optymalnych ścieżek stanowi kluczowy element efektywnego zarządzania siecią i przepływem danych w niej. Z naszych doświadczeń wynika, że kiedy te ścieżki są prawidłowo określone, klienci mogą cieszyć się znacznym wzrostem przepustowości przesyłu danych za pomocą protokołu UDP. Optymalnie wyznaczone trasy pozwalają na lepsze wykorzystanie dostępnych zasobów sieciowych, minimalizują opóźnienia oraz redukują ryzyko przeciążenia, co z kolei przekłada się na znacznie wydajniejszą komunikację i zadowolenie użytkowników z jakości usług sieciowych.

3.2 TCP

Podobnie jak w UDP, wybraliśmy również inne połączenie, które jest praktycznie identyczne z wcześniej rozważanym, czyli h7-h5. Jedyna różnica polega na tym, że dane w pierwszym połączeniu muszą przejść przez łącze **h2-s2-s4-s5-h5**, a w drugim **h7-s7-s4-s5-h5**. Na samym początku zaczęliśmy od badania przepustowości sieci przy jednym połączeniu dla domyślnych wartości kontrolera Minineta. Uzyskaliśmy następujące wyniki (każde o przepustowości łącza 10 Mb/s):

- Dla h2-h5 - **7,77 Mb/s**.
- Dla h7-h5 - **7,53 Mb/s**.

Następnie odpaliliśmy połączenia równolegle. Przepustowości jakie otrzymaliśmy były:

- Parametr max_bit_rate:
  - Dla tych samych parametrów nie przekraczających przepustowości łącza - **3,83 Mb/s** (h2-h5 -b 5M), **3,52 Mb/s** (h7-h5 -b 5M).
  - Dla tych samych parametrów przekraczających przepustowość łącza - **4,10 Mb/s** (h2-h5 -b 10M), **4,47 Mb/s** (h7-h5 -b 10M).
  - Dla różnych parametrów nie przekraczających przepustowości łącza - **1,49 Mb/s** (h2-h5 -b 2M), **1,40 Mb/s** (h2-h5 -b 2M), **1,62 Mb/s** (h7-h5 -b 2M), **1,47 Mb/s** (h7-h5 -b 2M).
  - Dla różnych parametrów przekraczających przepustowość łącza - **1,85 Mb/s** (h2-h5 -b 10M), **1,83 Mb/s** (h2-h5 -b 5M), **2,53 Mb/s** (h7-h5 -b 10M), **2,18 Mb/s** (h7-h5 -b 5M).
- Dla innych parametrów:
  - Różny parametr volume (4 połączenia równległe) - **1,90 Mb/s** (h2-h5 -n 5M), **2,76 Mb/s** (h2-h5 -n 10M), **2,03 Mb/s** (h7-h5 -n 5M), **2,38 Mb/s** (h7-h5 -n 10M).
  - Różny parametr window (4 połączenia równległe) - **2,26 Mb/s** (h2-h5 -w 100k), **1,96 Mb/s** (h2-h5 -w 50k), **2,28 Mb/s** (h7-h5 -w 100k), **1,86 Mb/s** (h7-h5 -w 50k).

Potem przystąpiliśmy do określenia optymalnych tras w celu usprawnienia połączenia przy jednoczesnym wykorzystaniu nowej topologii. Nowymi trasami są:

- **h2-h5** - h2-s2-s5-h5
- **h7-h5** - h7-s7-s5-h5

Nasze testy ponownie rozpoczęliśmy od badania przepustowości podczas pojedynczego połączenia (każde o przepustowości łącza 10 Mb/s):

- Dla h2-h5 - **8,00 Mb/s**.
- Dla h7-h5 - **8,03 Mb/s**.

Następnie odapliliśmy oba połączenia równolegle. Podobnie jak w UDP, z racji, że połączenia nie dzieliły ze sobą ścieżek, przepustowość ich łącza wzrosła dwukrtonie. Przepustowości jakie otrzymaliśmy były:

- Parametr max_bit_rate:
  - Dla tych samych parametrów nie przekraczających przepustowości łącza - **8,88 Mb/s** (h2-h5 -b 10M), **8,91 Mb/s** (h7-h5 -b 10M).
  - Dla tych samych parametrów przekraczających przepustowość łącza - **7,95 Mb/s** (h2-h5 -b 20M), **8,77 Mb/s** (h7-h5 -b 20M).
  - Dla różnych parametrów nie przekraczających przepustowości łącza - **1,49 Mb/s** (h2-h5 -b 2M), **1,40 Mb/s** (h2-h5 -b 2M), **1,62 Mb/s** (h7-h5 -b 2M), **1,47 Mb/s** (h7-h5 -b 2M).
  - Dla różnych parametrów przekraczających przepustowość łącza - **5,74 Mb/s** (h2-h5 -b 10M), **4,23 Mb/s** (h2-h5 -b 5M), **4,82 Mb/s** (h7-h5 -b 10M), **4,23 Mb/s** (h7-h5 -b 5M).
- Dla innych parametrów:

- Różny parametr volume (4 połączenia równległe) - **4,77 Mb/s** (h2-h5 -n 5M), **6,07 Mb/s** (h2-h5 -n 10M), **3,67 Mb/s** (h7-h5 -n 5M), **6,13 Mb/s** (h7-h5 -n 10M).
- Różny parametr window (4 połączenia równległe) - **4,68 Mb/s** (h2-h5 -w 100k), **3,96 Mb/s** (h2-h5 -w 50k), **4,03 Mb/s** (h7-h5 -w 100k), **3,89 Mb/s** (h7-h5 -w 50k).

Wszystkie doświadczenia powtórzyliśmy ponownie dla połączenia h3-h7. Oryginalna droga jaką musiało pokonać te połączenie to **h3-s3-s1-s2-s4-s7-h7** oraz podobna droga, którą musieliśmy wybrać z takich samych, jak w przykładzie powyżej względów, **h6-s6-s3-s1-s2-s4-s7-h7**, zaś optymalne drogi jakie wybraliśmy to były: dla pierwszej **h3-s3-s4-s7-h7**, dla drugiej **h6-s6-s4-s5-s7-h7**. Powtarzając te eksperymenty doszliśmy jednak do takich samych wniosków.

3.2.1 Wnioski dla TCP

Z naszych badań wynika, że optymalne trasy i zarządzanie siecią nie tylko korzystnie wpływają na protokół UDP, ale również mogą znacząco polepszyć wydajność przesyłu danych za pomocą protokołu TCP. Optymalne trasy i odpowiednie zarządzanie siecią mogą pomóc w minimalizowaniu tych czynników, co z kolei przekłada się na lepszą wydajność protokołu TCP. Dobre zarządzanie siecią może pomóc w redukcji opóźnień, poprawie przepustowości oraz zminimalizowaniu ryzyka utraty pakietów.

3.3 UDP a TCP

W poprzednim laboratorium sprawdzaliśmy połączenie h1-h10, które nie wymagało ingerencji innych switchów w te połączenie. Aby te testy miały sens, zmieniliśmy naszą drogę, rozszerzając ją o kilka switchy. Wybraliśmy więc drogę h10-h6, która oryginalnie miała ścieżkę **h10-s10-s1-s3-s6-h6** oraz h1-h6, która miała ścieżkę **h2-s2-s1-s3-s6-h6**, a ich optymalne wersje miały ścieżki **h10-s10-s8-s6-h6** oraz **h2-s2-s4-s6-h6**. Na początku ustaliliśmy identyczne parametry w obu połączeniach, jedynie zmieniając parametr dotyczący rozmiaru bloku danych.

Podobnie jak w 1 laboratorium, kiedy parametr -l miał niską wartość, podczas działania UDP przepustowość protokołu TCP była praktycznie zerowa, a w niektórych przypadkach wynosiła blisko zera. Dopiero po zakończeniu działania UDP protokół TCP zaczął działać normalnie.
Natomiast w przypadku większej wartości parametru -l straty pakietów w protokole TCP były znacznie mniejsze. To dlatego, że w przypadku, gdy UDP działa z dużą częstotliwością i małymi pakietami, może zdominować łącze sieciowe, zapełniając bufor i utrudniając efektywne działanie protokołu TCP. Dopiero po zakończeniu działania UDP protokół TCP może wrócić do normalnej pracy, gdy nie ma już konkurencji ze strony dużej liczby pakietów UDP.

Jednak w przypadku większego rozmiaru pakietów w protokole TCP, ten protokół jest w stanie przesyłać więcej danych w jednym pakiecie, co zmniejsza narzut na kontrolę przepływu i zwiększa efektywność przesyłania, nawet w obliczu jednoczesnej komunikacji UDP. Następnie powtórzyliśmy ten eksperyment dla optymalnych ścieżek, które nie dzieliły już tej samej ścieżki i uzyskaliśmy większe przepustowości na poszczególnych łączach.

3.4 Wnioski

Nasze badania wykazały, że optymalne trasy i skuteczne zarządzanie siecią mają korzystny wpływ nie tylko na protokół UDP, ale także mogą znacząco poprawić wydajność przesyłu danych za pomocą protokołu TCP. Optymalne trasy i właściwe zarządzanie siecią mogą pomóc w minimalizowaniu różnych czynników, które z kolei przekładają się na lepszą wydajność protokołu TCP.

Skuteczne zarządzanie siecią może przyczynić się do redukcji opóźnień, zwiększenia przepustowości oraz minimalizacji ryzyka utraty pakietów. Oznacza to, że poprawa infrastruktury sieciowej i optymalne trasy przesyłu danych mają istotne znaczenie nie tylko dla protokołu UDP, ale także dla TCP, co przekłada się na bardziej niezawodny i wydajny przesył danych w sieci.

4 Zadanie 3

Celem dostarczenia informacji dotyczącej sieci, utworzyliśmy własny format pliku. Zasady tworzenia pliku konfiguracyjnego:

- Nazwa musi być "net_file.txt"
- W pierwszym wierszu definiujemy hosty po przecinku, któremu każdemu będzie przydzielony jeden switch
- Zaczynając od trzeciego wiersza wypisujemy połączenia między switchami
- Połączenie między switchami definiujemy wypisując kolejno: numer switcha źródłowego, numer switcha docelowego, opóźnienie łącza, port switcha źródłowego do switcha dodelowego, port switcha dodelowego do switcha źródłowego, strata pakietów łącza oraz przepustowość łącza

Część przykładowego pliku konfiguracyjnego:

```
Paris,Toulouse,Lyon,Barcelona,Porto,Milan,Valencia,Frankfurt,Zurich,Bruges

1,2,4.16,2,2,0.1,10
1,3,2.76,3,2,1.1,10
1,10,1.9,4,2,2.5,10
3,6,2.40,3,2,3.2,10
4,2,1.79,2,3,0.1,10
4,5,6.37,3,2,0.1,10
...
```

Rysunek 8: Plik net_file.txt

Metody programu:

- Pliku graph.py:

  Inicjalizacja klasy `Graph`

  – Konstruktor (`__init__`) inicjalizuje graf z określoną liczbą wierzchołków (`vertices`).

  – Trzy macierze sąsiedztwa(`graph`, `graph_loss`, `graph_bd`) i słownik `graph_net` są używane do reprezentacji grafu, z różnymi celami (odległość, utrata, przepustowość i dodatkowe atrybuty).

  Metoda `adjacency_matrix_to_nx_graph`

  – Konwertuje macierz sąsiedztwa na graf NetworkX.

  Metoda `plot_whole_graph_and_path`

  – Rysuje cały graf i podkreśla wyliczoną ścieżkę.

  – Wykorzystuje NetworkX do utworzenia grafu i Matplotlib do wizualizacji.

  – Krawędzie są oznaczone informacjami o opóźnieniu, przepustowości i stracie pakietów na łączu.

  – Najkrótsza ścieżka jest wyróżniona na czerwono.

  Metoda `file_to_graph`

  – Odczytuje dane z pliku, konstruując macierze sąsiedztwa oraz związane z nią macierze na podstawie przepustowości łączy.

  Metoda `shortest_path`

  – Wykorzystuje algorytm Dijkstry z NetworkX do znalezienia najkrótszej ścieżki między dwoma wierzchołkami w grafie.

  – Zwraca ścieżkę jako ciąg znaków.

- Pliku controller.py:

  Metoda: `gui`

- Prosty interfejs wiersza polecenia dla kontrolera sieci.
- Udostępnia zestaw poleceń, takich jak "help", "connect", "exit"i "reset", które użytkownik może wprowadzić.
- Polecenie "connect"wymaga dodatkowych atrybutuów, by zażądać połączenie.

Metoda: `check_request`

- Sprawdza poprawność wprowadzonych danych żądanego połączenia przez użytkownika .
- Weryfikuje, czy dane zawierają poprawne słowa, liczby całkowite i mieszczą się w określonych zakresach.

Metoda: `net_graph`

- Odczytuje plik sieci i tworzy graf za pomocą klasy `Graph`.

Metoda: `reset_net`

- Nadpisuje plik, zawierajacy dane dotyczącego sieci i w jej trwajacych połączeń, plikiem konfiguracyjnym sieci tym samym resetując sieć.

Metoda: `create_connection`

- Tworzy połączenie sieciowe na podstawie określonych parametrów określonych przez klienta i pliku.
- Obejmuje znalezienie optymalnej ścieżki, określenie przepustowości i straty, interakcję z użytkownikiem oraz wysyłanie zasad do kontrolera sieci ONOS.

Metoda: `path_bd_and_loss`

- Oblicza przepustowość i straty dla podanej ścieżki sieci.

Metoda: `update_net`

- Aktualizuje sieć na podstawie utworzonego połączenia.

Metoda: `find_optimal_path`

- Określa optymalną ścieżkę dla połączenia sieciowego na podstawie określonych parametrów.

Metoda: `send_request`

- Wysyła żądanie do kontrolera sieci ONOS.

Metoda: `create_rules`

- Tworzy reguły konfiguracji switchy na podstawie określonego hosta źródłowego, hosta docelowego i ścieżki.

Metoda: `ports`

- Określa porty źródłowe i docelowe dla podanego switcha.

Metoda: `rule`

- Tworzy reguły dla switcha.

- Pliku main.py:

Metoda: `main`

- Uruchamia gui.

Program umożliwia użycie komend:

- help - Wypisuje wszystkie komendy programu
- connect - Uruchamia procedurę powstawania połączenia

- reset - Resetuje sieć do stanu początkowego dla podanego pliku konfiguracyjnego
- exit - Kończy działanie programu

Przedstawienie użycia komend "help", "reset"i "exit":

```
Write command
help
'connect' -  request connection
'exit' - exit program
'reset' - reset net
reset
Net reset
exit
Exiting program
```

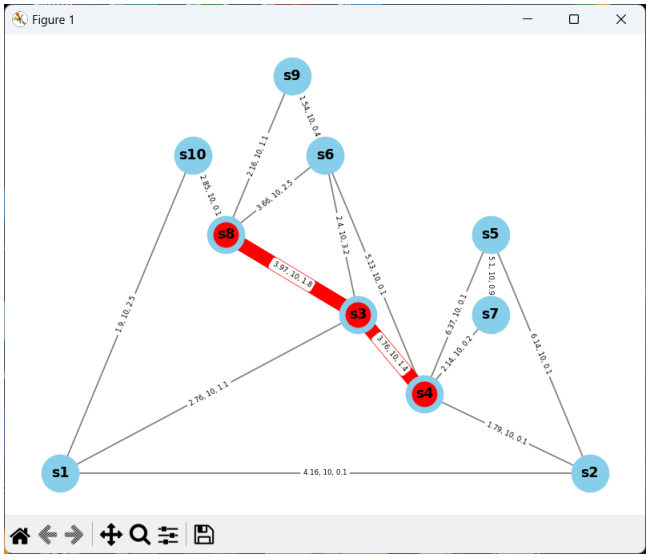
Rysunek 9: Komendy dostępne do użytku

Przedstawienie przykładowego użycia komendy "connect":

```
Write command
connect 4 8 tcp 5 _
Creating connection
path: HOST -> 4 -> 3 -> 8 -> HOST
loss: 3.2, bandwidth: 5
Do you want to create connection? Y/n: y
Response Switch11: {"flows":[{"deviceId":"of:000000
Connection created
exit
Exiting program
```

Rysunek 10: Wykonany program

W wyniku wykonania programu powstała ścieżka spełniająca podane parametry i mająca najmniejsze opóźnienie:



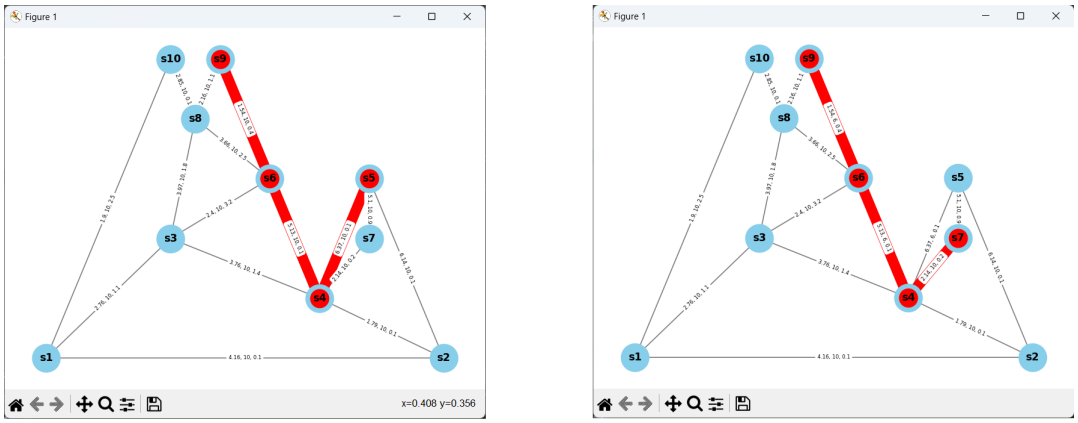
Rysunek 11: Ścieżka dla wykonanego programu

Gui pokaże dla połączenia UDP możliwe straty pakietów na ścieżce, jeżeli przepustowość ścieżki będzie mniejsza niż przepustowość żądana przez klienta. W przypadku protokołu TCP trudno zauważyć wpływ strat, ponieważ dla tego typu połączenia utrata pakietów prowadzi do spowolnienia transmisji. Natomiast w przypadku UDP, jeśli nadawane pakiety przekraczają przepustowość łącza, istnieje ryzyko utraty pakietów.

5 Zadanie 4

5.1 UDP

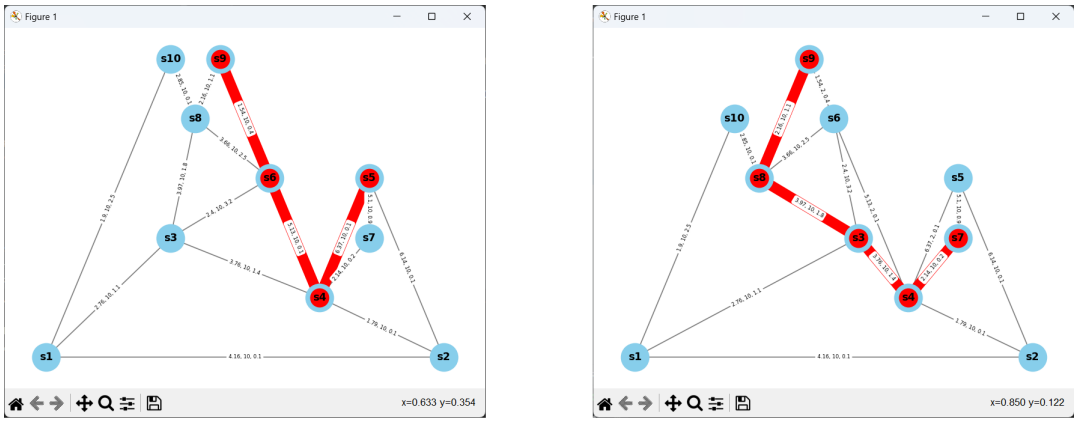
Na samym początku odpaliliśmy skrypt dla naszych dróg h5-h9 oraz h7-h9, ponieważ te drogi również testowaliśmy w zadaniu 2, ale mają bardziej rozwinięte ścieżki, co będzie lepiej się prezentowało na grafach. Pierwszym testem była sytuacja, gdy obie ścieżki nie przekraczały parametru przepustowości i wynosił on po 4M. Drogi jakie otrzymaliśmy wyglądały w następujący sposób:



Rysunek 12: Grafy h5-h9 i h7-h9 o parametrach -b 4M dla UDP

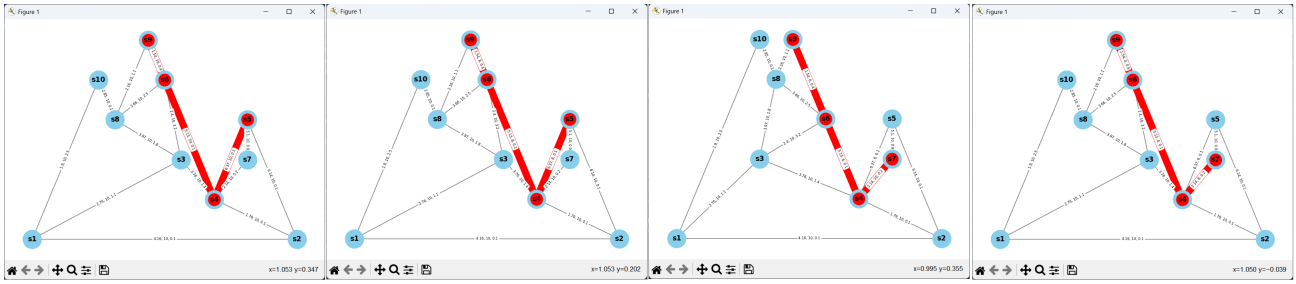
Dla drogi h5-h9 uzyskaliśmy wynik 4,18 Mb/s, dla h7-h9 4,18 Mb/s. Nie uświadczyliśmy strat pakietów.

Następnym doświadczeniem była sytuacja, gdy parametr przepustowości był taki sam, ale sumarycznie przekraczał on wartość 10 Mb/s.



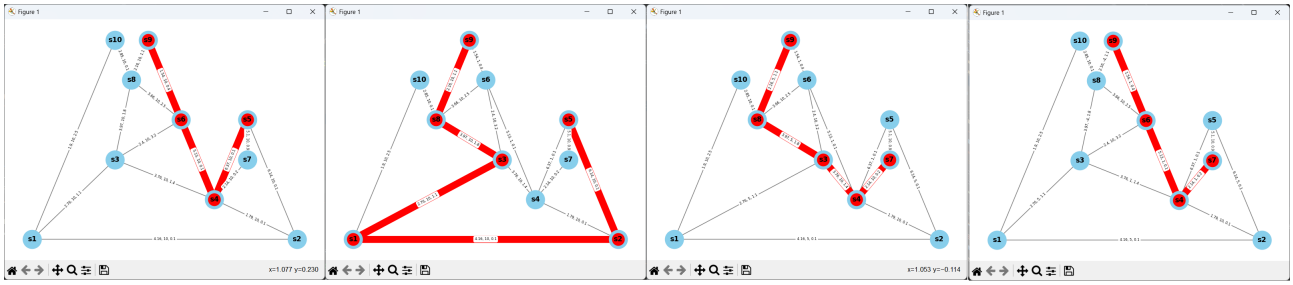
Rysunek 13: Grafy h5-h9 i h7-h9 o parametrach -b 8M dla UDP

Dla drogi h5-h9 uzyskaliśmy wynik 8,37 Mb/s, dla h7-h9 8,35 Mb/s. Nie uświadczyliśmy strat pakietów. Następnym testem było sprawdzenia jaka droga zostanie znaleziona przy 4 równoległych połączeniach, każde o przepustowości 2 Mb/s, czyli wartości, która nie będzie przekraczała przepustowości łącza.



Rysunek 14: Grafy h5-h9 i h7-h9, każde po 2 połączenia o parametrach -b 2M dla UDP

Tak jak się spodziewaliśmy, drogi zostały identycznie powielone tak jak w teście pierwszym, ponieważ sumarycznie nie przekraczały one przepustowości łącza, więc te same ścieżki mogły zostać powtórzone. Wyniki przepustowości, jakie otrzymaliśmy wynosiły: 2,09 Mb/s (h5-h9 -b 2M), 2,00 Mb/s (h5-h9 -b 2M), 2,09 Mb/s (h7-h9 -b 2M), 2,09 Mb/s (h7-h9 -b 2M). 2M), nie zaobserwowaliśmy strat pakietów. Ostatnim eksperymentem było zbadanie sytuacji, w której sumaryczna wartość przepustowści ścieżek przekracza wartość przepustowości łącza. Parametry -b ustawiliśmy w następujący sposób: 1 połączenie - h5-h9 -b 9M, 2 połączenie - h5-h9 -b 5M, 3 połączenie - h7-h9 -b 9M, 4 połączenie - h7-h9 -b 5M.

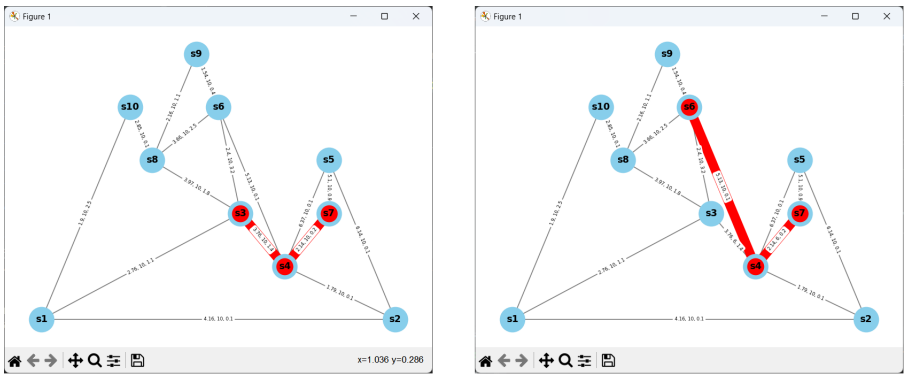


Rysunek 15: Grafy 1 połączenie - h5-h9 -b 9M, 2 połączenie - h5-h9 -b 5M, 3 połączenie - h7-h9 -b 9M, 4 połączenie - h7-h9 -b 5M dla UDP

Wyniki jakie otrzymaliśmy dla poszczególnych połączeń wynosiły - 5,45 Mb/s (h5-h9 -b 9M), 3,19 Mb/s (h5-h9 -b 5M), 6,12 Mb/s (h7-h9 -b 9M), 3,48 Mb/s (h7-h9 -b 5M). Podzczas próby zainicjowania połączenia (h7-h9 -b 9M) w terminalu otrzyamliśmy powiadomienie, że straty pakietów mogą osiągnąć duże wartości, ponieważ łącze było już wykorzystywane przez inne połączenie i przepustowość łącza nie byłą wystarczająca, aby obsłużyć obu klientów bez strat. Ostatecznie wszystkie połączenia miały straty na poziomie około 25%.
Wszystkie testy powtórzyliśmy również dla naszych oryginalnych ścieżek - **h3-h8** i **h9-h8**, jednak wnioski pozostały takie same.

5.2 TCP

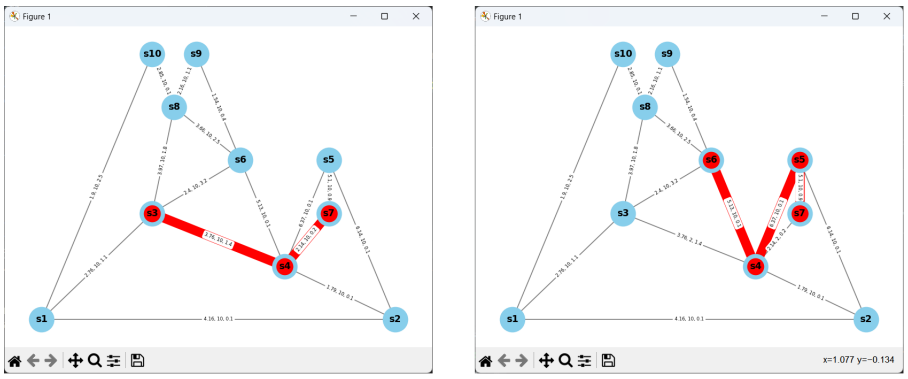
Nasze testy rozpoczęliśmy od włączenia skryptu dla naszych dróg h3-h7 oraz h6-h7, ponieważ podobnie jak dla UDP, te drogi testowaliśmy w zadaniu 2, ale mają bardziej rozwinięte ścieżki, co będzie lepiej się prezentowało na grafach. Pierwszym testem była sytuacja, gdy obie ścieżki nie przekraczały parametru przepustowości i wynosił on po 4M. Drogi jakie otrzymaliśmy wyglądały w następujący sposób:



Rysunek 16: Grafy h3-h7 i h6-h7 o parametrach -b 4M dla TCP

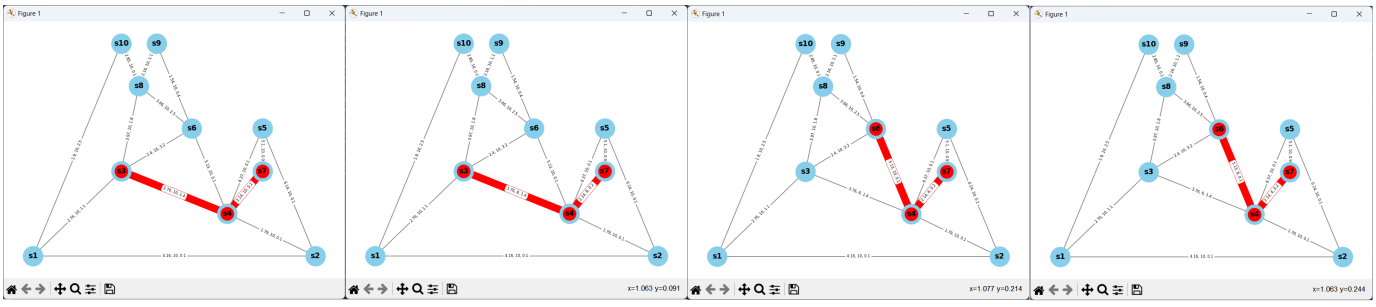
Dla drogi h3-h7 uzyskaliśmy wynik 3,64 Mb/s, dla h6-h7 3,60 Mb/s.

Następnym doświadczeniem była sytuacja, gdy parametr przepustowości był taki sam, ale sumarycznie przekraczał on wartość 10 Mb/s, więc ustawliśmy jego wartość na -b 8M. Oto jakie ścieżki zostałe wybrane:



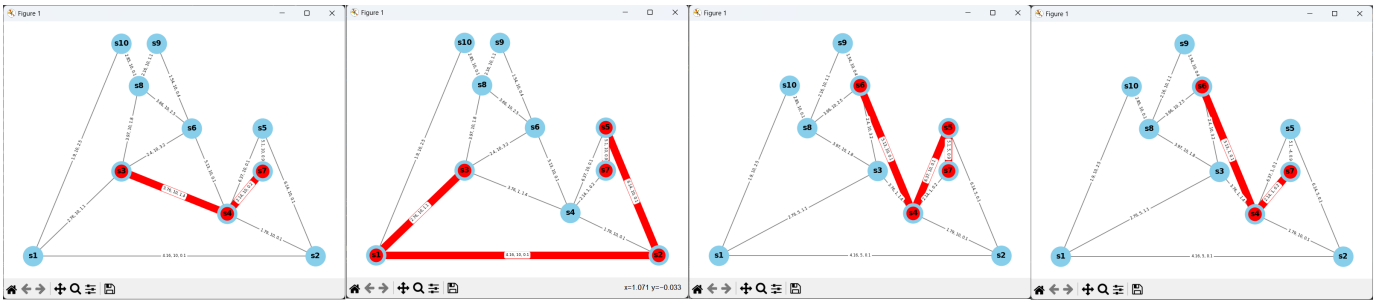
Rysunek 17: Grafy h3-h7 i h6-h7 o parametrach -b 8M dla TCP

Dla drogi h3-h7 uzyskaliśmy wynik 8,15 Mb/s, dla h6-h7 7,95 Mb/s. Następnym testem było sprawdzenia jaka droga zostanie znaleziona przy 4 równoległch połączeniach, każde o przepustowości 2 Mb/s, czyli wartości, która nie będzie przekraczała przepustowości łącza.



Rysunek 18: Grafy h3-h7 i h6-h7, 2 połączenia każde o parametrach -b 2M dla TCP

Podobnie jak w UDP, drogi zostały identycznie wyznaczone tak jak w teście pierwszym, ponieważ sumarycznie nie przekraczały one przepustowości łącza, więc te same ścieżki mogły zostać powtórzone. Wyniki przepustowości, jakie otrzymaliśmy wynosiły: 1,59 Mb/s (h3-h7 -b 2M), 1,25 Mb/s (h3-h7 -b 2M), 1,41 Mb/s (h6-h7 -b 2M), 1,30 Mb/s (h6-h7 -b 2M). Kolejnym eksperymentem było zbadanie sytuacji, w której sumaryczna wartość przepustowści ścieżek przekracza wartość przepustowości łącza. Parametry -b ustawiliśmy w następujący sposób: 1 połączenie - h3-h7 -b 9M, 2 połączenie - h3-h7 -b 5M, 3 połączenie - h6-h7 -b 9M, 4 połączenie - h6-h7 -b 5M.



Rysunek 19: Grafy 1 połączenie - h3-h7 -b 9M, 2 połączenie - h3-h7 -b 5M, 3 połączenie - h6-h7 -b 9M, 4 połączenie - h6-h7 -b 5M dla TCP

Wyniki jakie otrzymaliśmy dla poszczególnych połączeń wynosiły - 4,18 Mb/s (h3-h7 -b 9M), 3,83 Mb/s (h3-h7 -b 5M), 4,51 Mb/s (h6-h7 -b 9M), 4,01 Mb/s (h6-h7 -b 5M).

Wszystkie testy powtórzyliśmy również dla naszych oryginalnych ścieżek - **h2-h5** i **h7-h5**, jednak wnioski pozostały takie same.

5.3 Wnioski

Przeprowadzenie testów i potwierdzenie poprawności działania programu do wyszukiwania optymalnych ścieżek dla połączeń UDP i TCP jest kluczowym krokiem w zapewnieniu skuteczności i niezawodności systemu. Wyniki testów stanowią potwierdzenie, że program spełnia założone wymagania i jest gotowy do użycia w naszej sieci.

Poprawność działania tego rodzaju programu jest szczególnie istotna w kontekście zapewnienia efektywnego przesyłania danych poprzez protokoły UDP i TCP, które są powszechnie używane w sieciach komputerowych. Optymalne ścieżki pozwalają zoptymalizować wydajność połączeń, co przekłada się na lepsze doświadczenia użytkowników oraz efektywne wykorzystanie zasobów sieciowych.

6 Podsumowanie

Laboratorium dostarczyło nam cennego doświadczenia i głębszego zrozumienia roli, jaką pełni kontroler ONOS w dziedzinie zarządzania siecią. Centralne sterowanie i monitorowanie elementami sieciowymi przez ONOS stanowi kluczowy element w kierowaniu infrastrukturą sieciową. Pozwala to na skuteczne zarządzanie zasobami oraz usługami.

Dzięki zdobytym umiejętnościom w tworzeniu aplikacji do sterowania siecią, staliśmy się bardziej kompetentni w dostosowywaniu i optymalizowaniu działania naszej infrastruktury sieciowej. Możemy teraz projektować i implementować narzędzia, które są dostosowane do konkretnych potrzeb naszej sieci, co przekłada się na efektywność oraz lepsze dostosowanie do specyficznych wymagań projektów.

Ta wiedza jest nie tylko teoretyczna, ale również praktyczna, co otwiera przed nami szereg możliwości w zakresie zaawansowanych projektów związanych z nowoczesnymi technologiami sieciowymi. Możemy skutecznie przyczynić się do rozwoju i utrzymania nowoczesnej infrastruktury sieciowej, co jest kluczowe w erze dynamicznych wymagań związanych z przesyłaniem danych, łącznością i bezpieczeństwem sieciowym.

Podsumowując, laboratorium dostarczyło nam nie tylko teoretycznej wiedzy, ale również praktycznych umiejętności, które są bezpośrednio zastosowalne w projektach związanych z zaawansowanymi technologiami sieciowymi, pozwalając nam lepiej zrozumieć, jak skutecznie zarządzać współczesnymi sieciami komputerowymi.