Akademia Górniczo-Hutnicza
Wydział Informatyki, Elektroniki i Telekomunikacji
Kierunek Teleinformatyka

Jakość Obsługi w Sieciach

Projekt 1

Badanie parametrów QoS w sieciach SDN z wykorzystaniem platformy POX oraz emulatora Mininet

Justyna Gręda, Jan Ściga, Hubert Majdański

Kraków, 26 kwietnia 2023

Spis treści

1 Wstęp

Opracowanie: Justyna Gręda

Wraz ze wzrostem liczby osób korzystających z Internetu na całym świecie zaczęły pojawiać się usługi przesyłające bardzo konkretnych ruch sieciowy, takie jak *streaming*, wideo konferencje czy *gaming*. Sprawiło to, że konieczne były badania nad jakością świadczonych usług (*Quality of Service - QoS*) i dążeniem do zapewnienia użytkownikom końcowym jak najwyższego ich poziomu. QoS może być mierzony za pomocą podstawowych metryk sieci, takich jak: strata pakietów (*packet loss ratio*), opóźnienie na łączu (*delay*), zmiana opóźnienia (*jitter*) czy przepływność (*throughput*) [10]. Niektóre usługi, takie jak przesyłanie głosu nie pozwalają na utratę pakietów, ponieważ przy zbyt wysokim *packet loss ratio* informacja staje się niezrozumiała dla użytkownika końcowego. Ruch w sieci musi być zatem odpowiednio dostosowany do wszystkich typów przesyłanych danych.

Przy rosnących potrzebach użytkowników coraz popularniejsze stają się sieci zdefiniowane programowo - *Software Defined Networks* (SDN). Pozwalają one oddzielić płaszczyznę sterowania (*control plane*) od płaszczyzny danych (*data plane*), co czyni sieć bezpieczniejszą i lepiej zarządzaną [2]. Najważniejszym elementem sieci SDN jest kontroler, który steruje pracą całej sieci z pomocą odpowiedniego protokołu. Jest to możliwe dzięki scentralizowanej budowie kontrolera, która pozwala mu poznać pełną topologię sieci [5]. Komunikuje się on poprzez protokół z przełącznikami i instruuje je, jak przełączać ruch. Najpopularniejszym protokołem SDN jest OpenFlow. W pracy przedstawiony zostanie POX - framework wykorzystujący do komunikacji między kontrolerem a przełącznikami protokół OpenFlow lub Open vSwitch Database Management Protocol (OVSBD).

Aby zapewnić jak najwyższy poziom QoS w sieciach SDN przedmiotem badań ekspertów jest między innymi odpowiednie położenie kontrolera w sieci czy liczba przełączników [6]. Istotny jest także odpowiednio dobrany do potrzeb ruchu protokół, np. OpenFlow pozwala sterować ruchem na poziomie przepływów [14]. W ramach projektu porównane zostaną podstawowe metryki QoS w sieciach SDN zbadane przez kilka narzędzi, takich jak Iperf, D-ITG (omówiony szerzej w rozdziale 3) czy ping. Do badań użyte zostaną predefiniowane w emulatorze Mininet trzy topologie SDN - *Single*, *Linear* oraz *Tree*. W dalszej części pracy podjęty zostanie temat zaawansowanych funkcji generatora ruchu D-ITG.

2 Przegląd wykorzystanych narzędzi SDN

Opracowanie: Hubert Majdański

2.1 Mininet

Mininet jest emulatorem sieci pozwalającym na naśladowanie sieci składających się z wielu hostów, przełączników oraz znajdujących się między nimi połączeń korzystając z jednej maszyny. Posiada on zarówno interfejs graficzny (GUI) jak i tekstowy (CLI). Mininet do tworzenia emulowanych sieci wykorzystuje opartą na procesach wirtualizację wspieraną przez jądro systemu Linux. Każdy z procesów posiada własny interfejs, dzięki czemu może korzystać jedynie z własnych procesów oraz może wykonywać kod tak jak normalny serwer korzystający z systemu Linux [12]. Emulator ten pozwala również na ustalenie topologii sieci, zarówno korzystając z jednej z wbudowanych topologii (jak wykorzystane w tej pracy *Single*, *Linear* oraz *Tree*), jak i tych bardziej rozbudowanych przy pomocy skryptów w języku Python Pozwala to na szybkie sprawdzenie jakości działania kontrolera w różnych środowiskach. Zaletą oprogramowania Mininet jest również łatwe skalowanie rozmiarów sieci w celu oceny zachowania kontrolera w zależności od jej wielkości [13]. Wykorzystanie emulatora Mininet oraz sieci sterowanych programowo pozwala na dużo łatwiejsze testowanie oraz projektowanie niż w przypadku klasycznych sieci. Zaprojektowane w ten sposób kontrolery oparte o protokół OpenFlow mogą zostać w łatwy sposób zaimplementowane fizycznie. W trakcie rozwoju sieci przy pomocy emulatora w celu lepszej oceny jakości jej działania używane są takie narzędzia jak Wireshark czy dpctl [12]. Przy pomocy *traffic control* (TC) wbudowanego w system operacyjny Linux połączenia między przełącznikami oraz hostami mogą być symulowane wraz z występującymi w nich opóźnieniami czy ograniczeniami przepływności, co pozwala na lepszą ocenę kontrolera w warunkach zbliżonych do rzeczywistych.

2.2 POX

POX jest kontrolerem SDN typu *Open Source* bazującym na protokole OpenFlow. Jest to kontroler oparty o scentralizowaną architekturę sieci [4]. POX jest następcą kontrolera NOX oferującym lepszą wydajność. Został on zaimplementowany przy użycia języka *Python*. Jego moduły korzystają z podstawowych funkcji języka Python oraz paru dodatkowych funkcji z bibliotek POX. Oficjalnie wspierane są systemy operacyjne *Windows*, *Mac OS* oraz *Linux*. Oprócz prostych elementów jak uczące się przełączniki, korzystając z środowiska POX można również implementować bardziej skomplikowane struktury jak równoważniki obciążenia czy zapory ogniowe [3]. Wraz z kontrolerem POX w repozytorium znajdują się przykładowe komponenty z możliwymi do ponownego użycia elementami, jak wybieranie ścieżki pakietu. Większość z nich może ze sobą współpracować, dzięki czemu bardziej skomplikowane kontrolery mogą być projektowane w prosty sposób, a dodatkowo znacząco ułatwia to dodawanie do kontrolera nowych funkcjonalności. Jedną z głównych zalet POX jest łatwość uruchomienia go oraz prostota języka Python pozwalająca na tworzenie własnych kontrolerów nie posiadając dużego doświadczenia programistycznego [3]. Kontroler POX wspiera również API ułatwiające rozwój SDN.

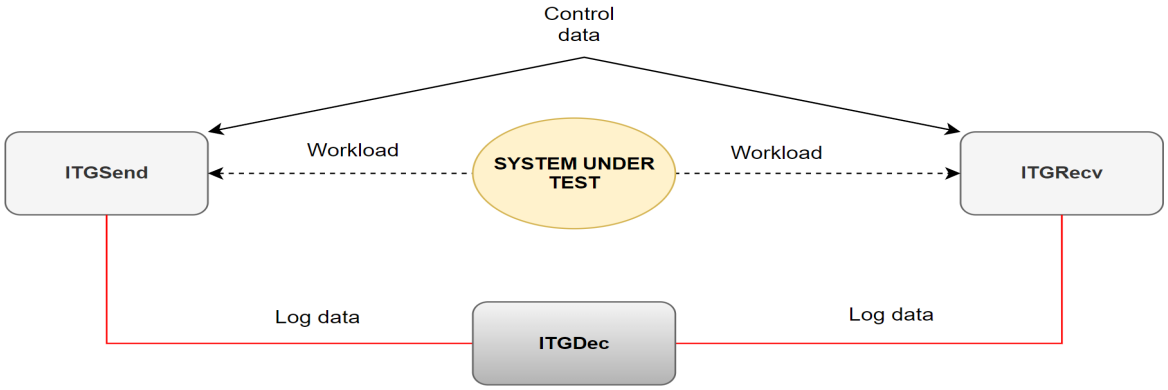
3 D-ITG jako zaawansowane narzędzie do generowania ruchu

Opracowanie: Jan Ściga

D-ITG (*Distributed Internet Traffic Generator*) to platforma umożliwiająca generowanie ruchu sieciowego odwzorowującego realne obciążenie aplikacji sieciowych stworzona przez naukowców z Uniwersytetu Neapolitańskiego [1]. D-ITG jest narzędziem wspierającym zarówno IPv4 jak i IPv6, potrafiącym imitować statystyczne parametry ruchu takich protokołów jak VOIP, DNS czy Telnet [9]. Narzędzie to jest dostępne dla wielu systemów operacyjnych takich jak Windows, Debian czy Fedora [11], wspiera również szereg rozkładów prawdopodobieństw używanych do modelowania ruchu sieciowego [7].

3.1 Schemat działania

Działanie generatora D-ITG ogranicza się do uruchomienia skryptu *ITGSend* imitującego klienta oraz *ITGRecv* imitującego serwer w komunikacji pomiędzy badanymi elementami sieci. Narzędzie *ITGDec* służy do zbierania i analizowania danych powstałych w wyniku symulacji sieciowej. Uproszczony schemat działania generatora zilustrowano na Rysunku 1.



Rysunek 1: Schemat działania generatora D-ITG

3.2 Wspierane metryki, protokoły oraz rozkłady prawdopodobieństwa

Generator D-ITG wspiera zaawansowane metryki *Quality of Service* mogące służyć do głębokiej analizy działania aplikacji lub sieci. Są to m.in. throughput, jitter, oneway-delay (OWD), round-trip-time (RTT), packet loss. Narzędzie pozwala emulować ruch popularnych protokołów sieciowych takich jak TCP, UDP, ICMP, DNS, Telnet czy VOIP. D-ITG posiada również możliwość generowania ruchu posiadającego zmienną wielkość pakietu. Wspieranymi rozkładami prawdopodobieństwa są rozkłady: Constant, Uniform, Exponential, Pareto, Cauchy, Normal, Poisson oraz Gamma.

3.3 Porównanie z innymi narzędziami oraz zastosowanie w badaniach naukowych

D-ITG wyróżnia się na tle innych narzędzi służących do generowania ruchu. Pośród innych dostępnych generatorów takich jak Iperf, Netperf czy IP Traffic nie można znaleźć narzędzia typu *Open Source* pozwalającego odtworzyć działanie tak wielu protokołów warstwy aplikacji czy transportu, rozkładów prawdopodobieństw oraz metryk badanego ruchu. Badane narzędzie jest szeroko wykorzystywane przez naukowców np. do badania odporności serwerów TCP na ataki typu *Denial of Service* [8].

	Iperf	Netperf	D-ITG	IP Traffic
Interface	Command line	Command line	Command line	GUI interface
Multi-platform	Yes	Yes	Yes	Windows only
User guide	Yes	Yes	Yes	Yes
Protocols	TCP and UDP	TCP, UDP SCTP, DLPI	TCP, UDP, ICMP, DNS, Telnet, VoIP	TCP, UDP IGMP
Packet departure Packet delay	No	No	Yes	Yes
Probability distributions	No	No	Yes	Yes
Log file	Yes	No	Yes	Yes
Internet Protocol	IPv6 and IPv4	IPv6 and IPv4	IPv6 and IPv4	IPv6 and IPv4
Measurements metrics	Jitter Packet loss Throughput	Packet loss Throughput Response time CPU usage	One-way-delay Round-trip-time Packet loss Jitter Throughput	Throughput Round trip time Packet loss Jitters

Tabela 2: Porównanie działania dostępnych generatorów ruchu [8]

4 Topologie symulacyjne

Opracowanie: Justyna Gręda, Jan Ściga, Hubert Majdański

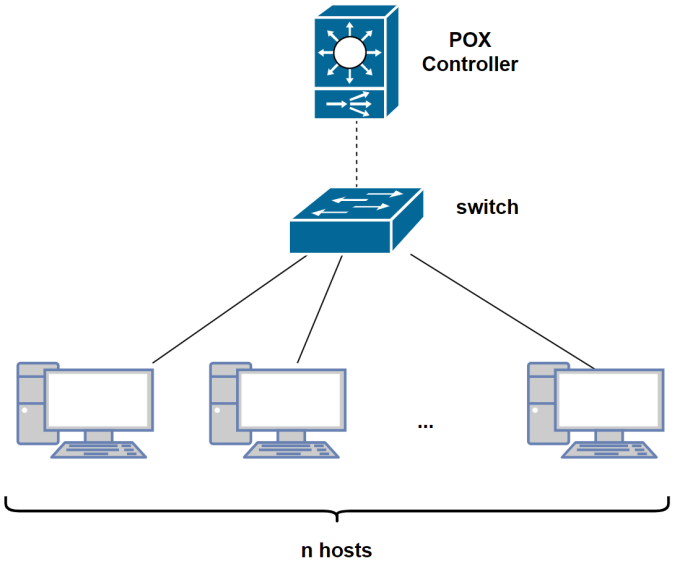
W toku badań symulacyjnych wykorzystane są topologie sieciowe, które zostały wcześniej zaimplementowane i udostępnione wraz z emulatorem Mininet. Uruchomienie danej topologii sprowadza się nie tylko do wywołania skryptu napisanego w języku Python wraz z odpowiednimi parametrami topologii (np. liczby przełączników, hostów), ale również atrybutami danej konfiguracji (opóźnienia danego połączenia lub ilości strat pakietów).

W projekcie badane są następujące topologie sieciowe:

1. Topologia Single
2. Topologia Linear
3. Topologia Tree

4.1 Topologia Single

Topologia *Single* jest najbardziej podstawową predefiniowaną w emulatorze Mininet topologią. Składa się ona z jednego kontrolera SDN (w projekcie używany jest POX), jednego przełącznika oraz n hostów. Przykładową topologię *Single*, która będzie badana przedstawiono na Rysunku 5.



Rysunek 3: Topologia *Single*

Tak jak pozostałe domyślne topologie, topologia *Single* może zostać uruchomiona z poziomu *command line*, z wykorzystaniem komendy przedstawionej w Listingu 1

```
wifi@wifi-VirtualBox:~$ sudo mn --topo single,n --controller=remote
```

Listing 1: Uruchomienie topologii single dla n hostów

4.2 Topologia Linear

Topologia *Linear* została zdefiniowana w emulatorze Mininet jako jedna z predefiniowanych topologii, którą można w prosty sposób uruchomić oraz skonfigurować z poziomu *command line*. Badana topologia składa się z:

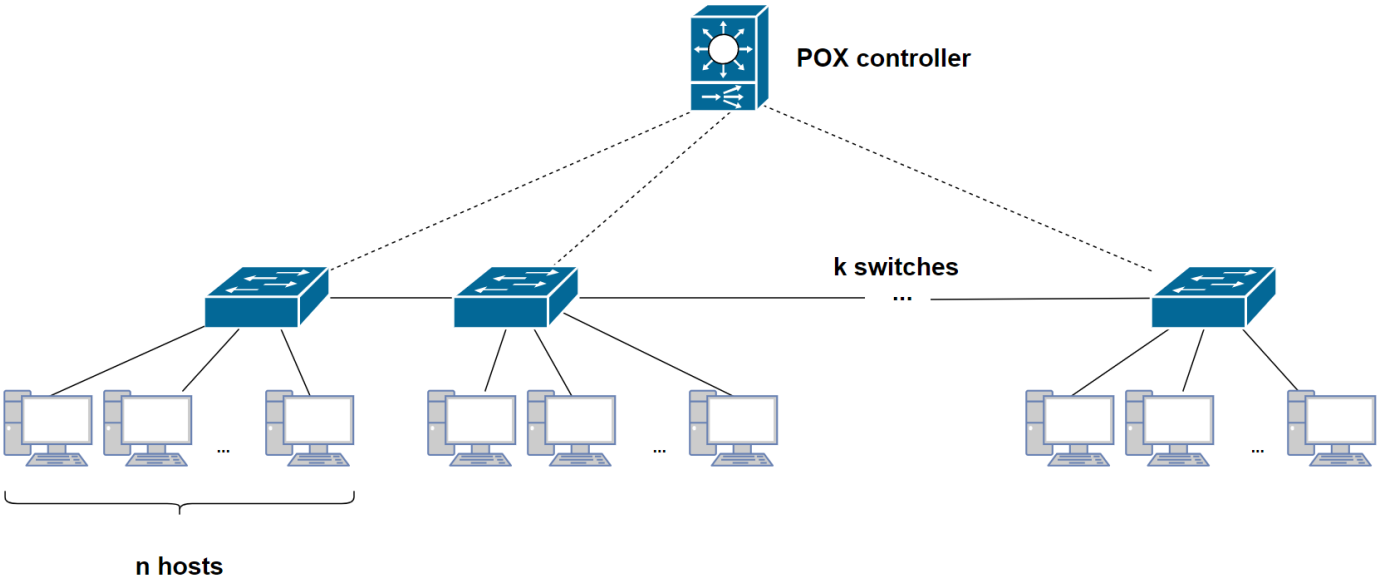
- Kontrolera SDN (POX)
- k połączonych ze sobą w sposób liniowy (*Linear*) przełączników
- n hostów dołączonych do każdego z przełączników

Liczbę hostów oraz przełączników ustala się z wykorzystaniem komendy uruchamiającej emulację sieci widocznej w Listingu 3

```
wifi@wifi-VirtualBox:~$ sudo mn --topo linear,k,n --controller=remote
```

Listing 2: Uruchomienie topologii linear dla k switchy oraz n hostów pod każdym ze switchy

Badaną topologię przedstawiono w sposób graficzny na Rysunku 4. Przedstawia on połączenia zdefiniowane pomiędzy transmitującymi węzłami oraz ilości konfigurowanych przełączników (k) oraz hostów (n).



Rysunek 4: Topologia *Linear*

4.3 Topologia Drzewa

Ostatnią z badanych przez nas topologii jest topologia drzewa. Jest ona predefiniowaną w emulatorze Mininet topologią, możliwą do uruchomienia za pomocą *command line*. Składa się ona z:

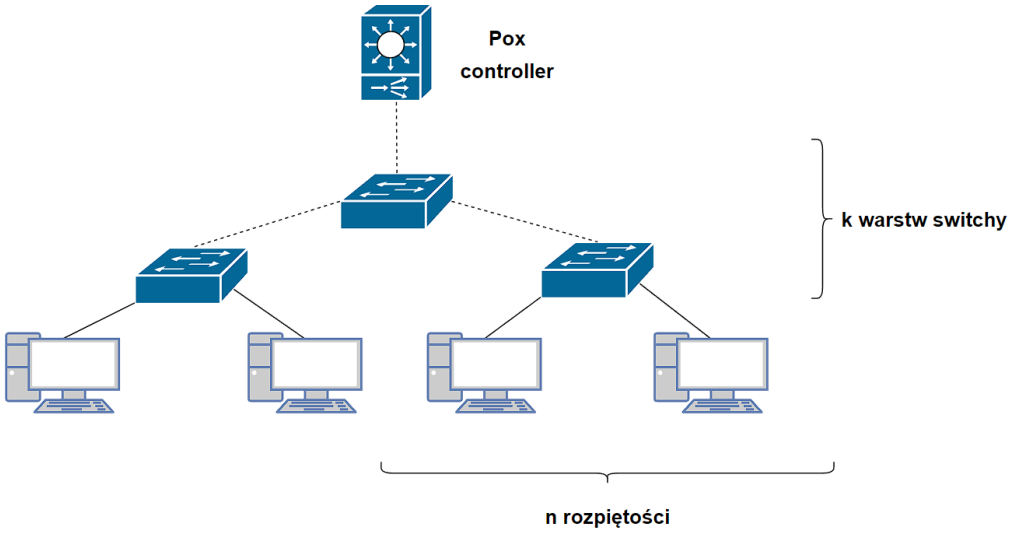
- Kontrolera SDN (POX)
- k warstw przełączników połączonych ze sobą w sposób rozgałęziający się *(tree)*
- n rozpiętości drzewa między warstwami

Za pomocą emulatora mininet możliwa jest ona do uruchomienia przy pomocy komendy:

```
wifi@wifi-VirtualBox:~$ sudo mn --topo tree,k,n --controller=remote
```

Listing 3: Uruchomienie topologii tree dla k warstw switchy oraz n rozpiętości drzewa

Topologię przedstawiono graficznie na Rysunku 5.



Rysunek 5: Topologia *Tree* z k warstwami przełączników oraz rozpiętością drzewa n

5 Wyniki symulacji

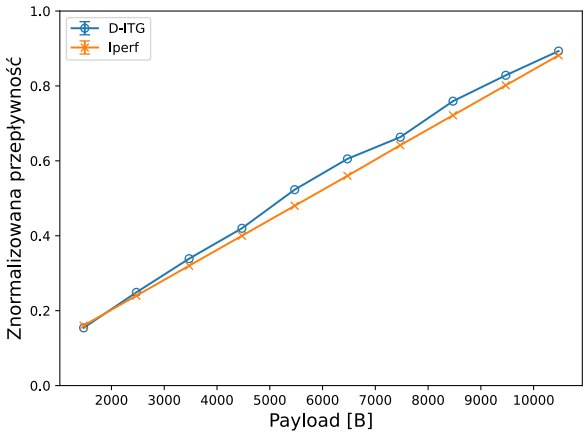
5.1 CZĘŚĆ 1: Analiza porównawcza generatorów ruchu

5.1.1 Topologia Single

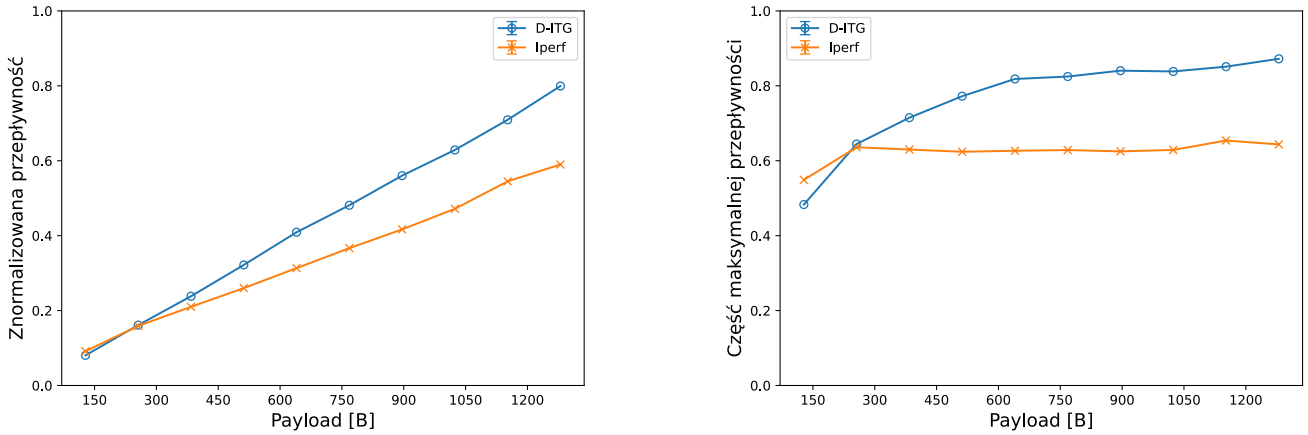
Opracowanie: Justyna Gręda

Scenariusz	Protokół	Liczba hostów (n)	Parametr na łączu	Wartość parametru
1	TCP	10	Opóźnienie	100 ms
2	UDP	20	Strata pakietów	20%

Tabela 6: Scenariusze symulacji topologii *Single*



Rysunek 7: Porównanie wyników dla scenariusza 1



(a) Przepływność

(b) Część maksymalnej przepływności

Rysunek 8: Porównanie wyników dla scenariusza 2

Przedstawienie konfiguracji: W ramach najprostszej oferowanej przez emulator Mininet topologii sprawdzono przepływność w zależności od wielkości przesyłanych danych (*Payload*). Dodatkowo, na łączu ustawione zostały parametry takie jak opóźnienie w scenariuszu pierwszym wynoszące 100 ms oraz 20 procentowa strata pakietów dla scenariusza drugiego. Scenariusze różnią się pomiędzy sobą wykorzystanymi protokołami, są to odpowiednio TCP i UDP oraz zakresem badanego *payloadu*: 1472-10472 B oraz 128-1280 B. Do badań wykorzystano dwa narzędzia: generator D-ITG oraz iperf.

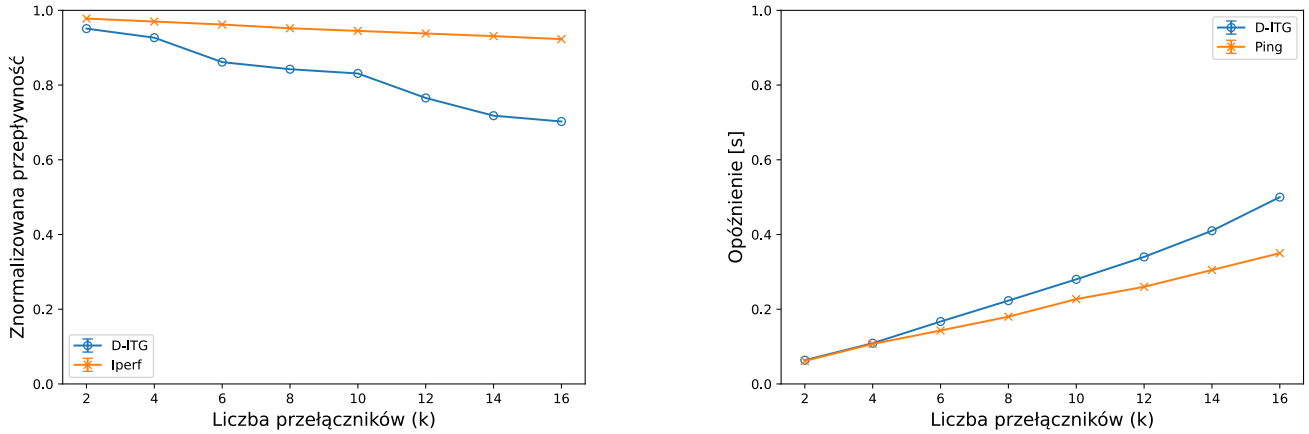
Omówienie wyników: W przypadku scenariusza 1 można zauważyć, że oba narzędzia dają bardzo porównywalne wyniki. Wraz ze zwiększającym się *payloadem*, rośnie także przepływność z powodu efektywniejszego wykorzystania kanału. Kształt obu krzywych wskazuje na zależność liniową. Dla scenariusza 2 sprawdzono także jak dużą część maksymalnej przepływności łącza udało się osiągnąć. W tym przypadku wyniki różnią się od siebie. Przepływność, która również rośnie wraz ze wzrostem *payloadu*, jest wyższa dla ruchu generowanego przez D-ITG. Ponadto, znacznie lepiej radził on sobie z 20-procentową stratą pakietów na łączu niż iperf, który dla większych porcji danych miał duże problemy z uruchomieniem się czy generowaniem informacji.

5.1.2 Topologia Linear

Opracowanie: Jan Ściga

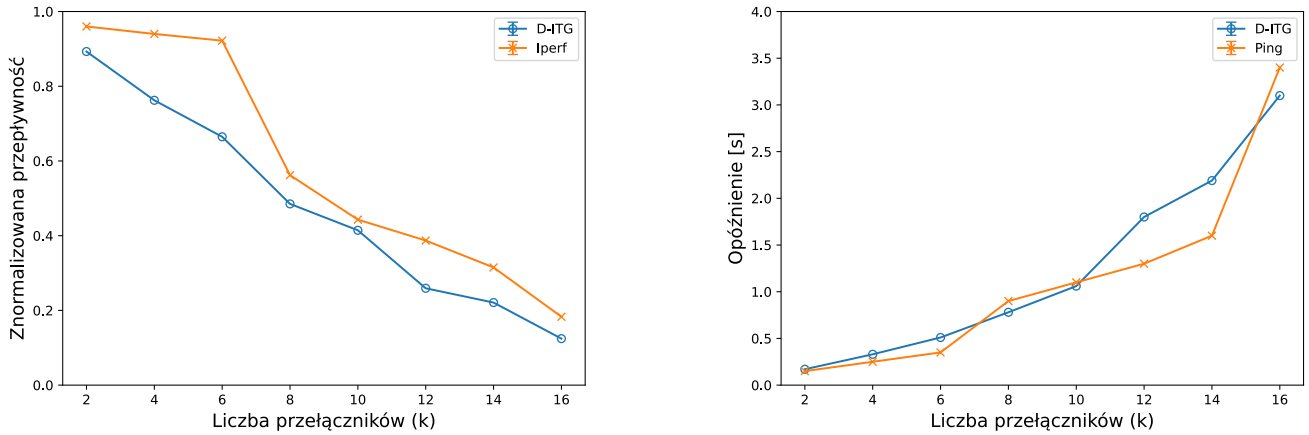
Scenariusz	Protokół	Opóźnienie łącza [ms]	Payload [B]	liczba hostów (n)
1	TCP	20	1472	2
2	TCP	50	1472	4

Tabela 9: Scenariusze symulacji topologii *Linear*



(a) Przepływność (b) Opóźnienie

Rysunek 10: Porównanie wyników dla scenariusza 1



(a) Przepływność (b) Opóźnienie

Rysunek 11: Porównanie wyników dla scenariusza 2

Przedstawienie konfiguracji: Symulacje podzielono na dwa scenariusze, w których zbadano podstawowe metryki QoS, jakimi są przepływność oraz opóźnienie w dziedzinie ilości przełączników obecnych w topologii *Linear* (patrz Rysunek 4). Scenariusz 1 przedstawiony na Rysunku 10 zakładał mniejsze opóźnienie (20 ms) na łączu niż Scenariusz 2 przeanalizowany na Rysunku 11 (50 ms). Badanymi narzędziami do generowania ruchu był generator D-ITG, który był porównywany z narzędziami ping oraz iperf.

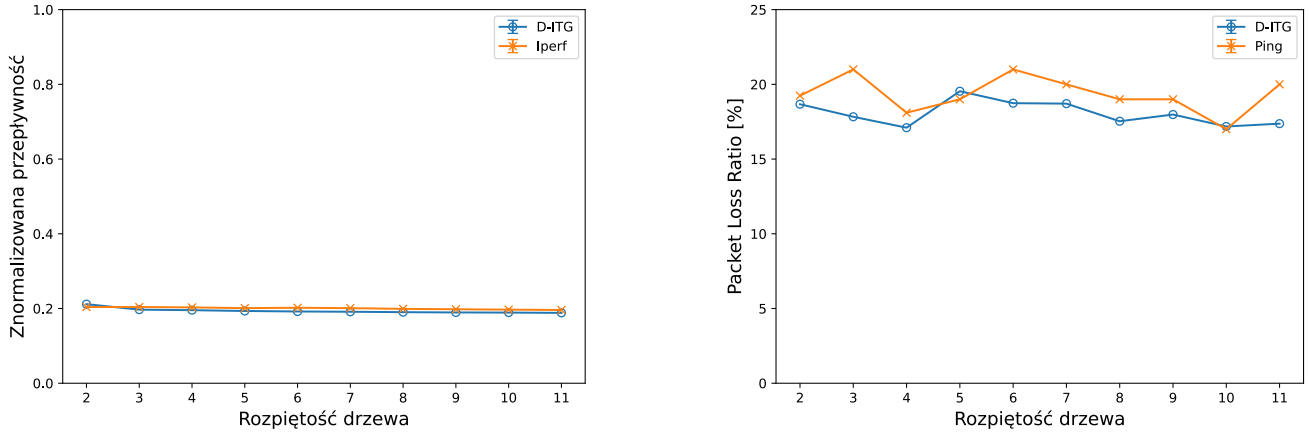
Omówienie wyników: Obydwa wykonane scenariusze symulacyjne pokazują, iż wzrost liczby przełączników w topologii *Linear* skutkuje spadkiem poziomu przepływności oraz wzrostem opóźnienia w kształcie zbliżonym do liniowego. Dodatkowo, w symulacji widocznej na Rysunku 11 w porównaniu do symulacji widocznej na Rysunku 10 widać wyraźnie, że spadek przepływności oraz wzrost opóźnienia jest znacznie bardziej stromy na skutek większego opóźnienia wprowadzanego przez pojedyncze łącze. Zarówno generator D-ITG, jak i iperf oraz narzędzie ping umożliwiają przeprowadzanie symulacji związanych z badaniem przepływności oraz opóźnienia i wykazują podobne właściwości, co zobrazowano na przykładzie powyższych scenariuszy symulacyjnych.

5.1.3 Topologia Drzewa

Opracowanie: Hubert Majdański

Scenariusz	Protokół	Parametr na łączu	Wartość parametru
1	UDP	Strata pakietów	5%
2	TCP	Opóźnienie	30 ms

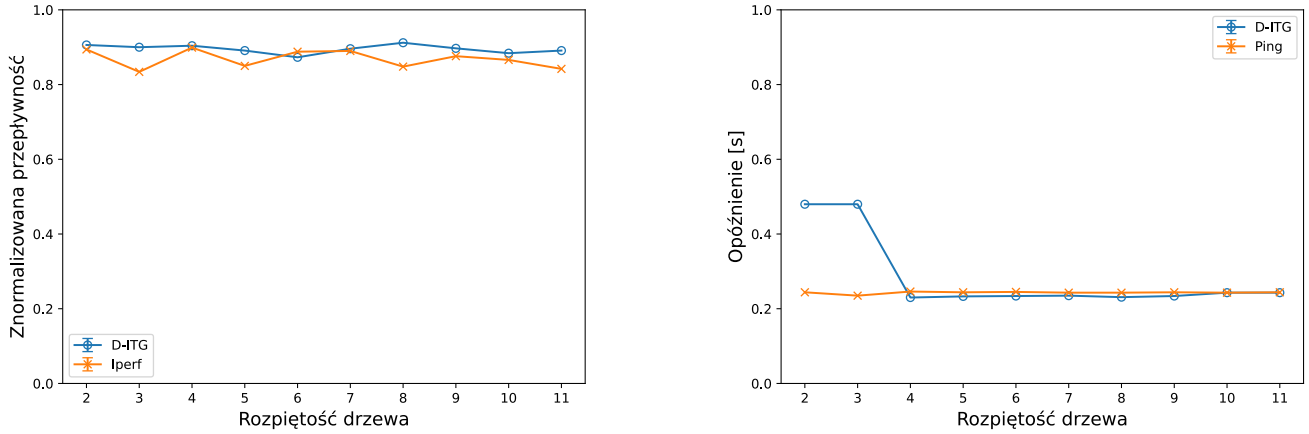
Tabela 12: Scenariusze symulacji topologii *Tree*



(a) Przepływność

(b) Packet Loss Ratio

Rysunek 13: Porównanie wyników dla scenariusza 1



(a) Przepływność

(b) Opóźnienie

Rysunek 14: Porównanie wyników dla scenariusza 2

Przedstawienie konfiguracji: Symulacja wykonana została dla dwóch scenariuszy - pierwszego, wykorzystującego protokół UDP, w którym strata pakietów na łączu wynosiła 5 % (Rysunek 13) oraz scenariusza drugiego, wykorzystującego protokół TCP oraz zakładającego opóźnienie na łączu 30ms (Rysunek 14). Symulacje zostały przeprowadzone dla drzewa o głębokości 2 oraz rozpiętości zmieniającej się w zakresie 2-11. W obu scenariuszach zostały wyznaczone metryki QoS - znormalizowana przepływność oraz współczynnik utraty pakietów *packet loss ratio* w przypadku scenariusza 1, oraz znormalizowana przepływność i opóźnienie w przypadku scenariusza 2. W celu generacji ruchu zostały wykorzystane narzędzia iperf, D-ITG oraz ping.

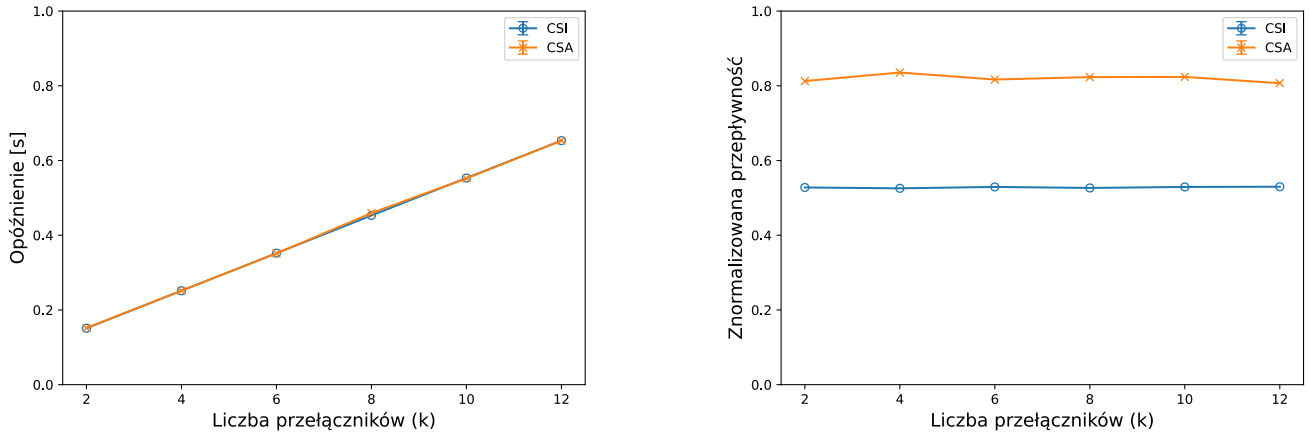
Omówienie wyników: W przypadku obu scenariuszy zauważyć można, iż mimo wzrostu rozpiętości drzewa - powodującej znaczący wzrost liczby hostów oraz przełączników, metryki QoS nie ulegają pogorszeniu. Wynika to z tego, iż wszystkie przełączniki są połączone za pomocą jednego przełącznika znajdującego się w warstwie wyższej, dzięki czemu droga pakietu jest krótsza. Pozwala to na stwierdzenie, iż topologia *Tree* jest odpowiednia w przypadku sieci obsługującej dużą liczbę hostów. Dodatkowo wartym zauważenia jest fakt, iż w przypadku scenariusza 1 wartość *packet loss ratio* wynosi jedynie 20% przy 5% spadku na łączu, co jest niską wartością, która nie byłaby możliwa do osiągnięcia przy użyciu prostszych topologii. Wszystkie z wykorzystanych generatorów danych działały poprawnie w obu scenariuszach.

5.2 CZĘŚĆ 2: Analiza zaawansowanych funkcji generatora D-ITG

5.2.1 Emulowanie ruchu sieciowego

Opracowanie: Justyna Gręda

W ramach sprawdzenia generatora D-ITG w bardziej rzeczywistych warunkach wykorzystano jego zdolność do emulowania specyficznego ruchu. Spośród kilku dostępnych opcji dla warstwy aplikacji wybrano odniesienie się do jakości obsługi w grach. D-ITG pozwala na generowanie ruchu specyficznego dla graczy Counter Strike w dwóch przypadkach - gdy pozostają aktywni w meczu (opcja *CSa*) lub nieaktywni, przebywają jedynie w grze (*CSi*). Zdecydowano się sprawdzić, jak zmiana topologii *Linear* wpłynie na opóźnienie w sieci przy ustawieniu opóźnienia 50 ms na łączu, a jednocześnie na poziom grywalności. Do generowania ruchu został wykorzystany protokół UDP. Wyniki symulacji ukazano na Rysunku 15.



(a) Opóźnienie

(b) Przepływność

Rysunek 15: Zestawienie wyników dla graczy CS

Opóźnienie dla obu przypadków jest niemal identyczne i zwiększa się liniowo w zależności od liczby przełączników. Różnią się one natomiast poziomem średniego *bitrate*. Gracz, który pozostaje aktywny w grze i rozgrywa mecz ma większy *bitrate* niż ten, który np. przegląda sklep czy zostawił włączone menu i odszedł od komputera, ponieważ generuje on więcej danych w konkretnej jednostce czasu. Z perspektywy zapewnienia jakości obsługi wymaga on także mniejszego opóźnienia niż gracz w stanie *idle*.

Następnie zbadano zachowanie emulowanego ruchu *Voice Over IP* (VoIP) w topologii *Linear*. Generator D-ITG pozwala na wybranie kilku parametrów dla ruchu typu VoIP, takich jak: Kodek (sprawdzono trzy z pięciu dostępnych), protokół transmisji dźwięku (*Real Time Protocol* - RTP, domyślny lub *Real Time Protocol with header compression* - CRTP), a także włączenie detekcji aktywności głosowej (*VAD*, domyślnie wyłączona). Ustawienia kodeków głosowych badanych w projekcie przedstawiono w Tabeli 16.

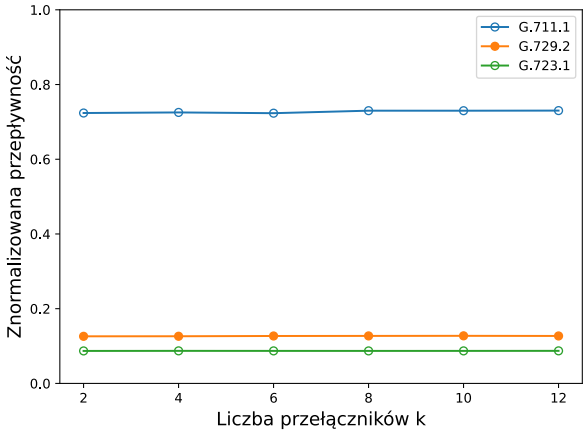
Kodek	*Framesize*	*Samples*	*Packets per sec*
G.711.1	80	1	100
G.729.2	10	2	50
G.723.1	30	1	26

Tabela 16: Parametry poszczególnych kodeków VoIP

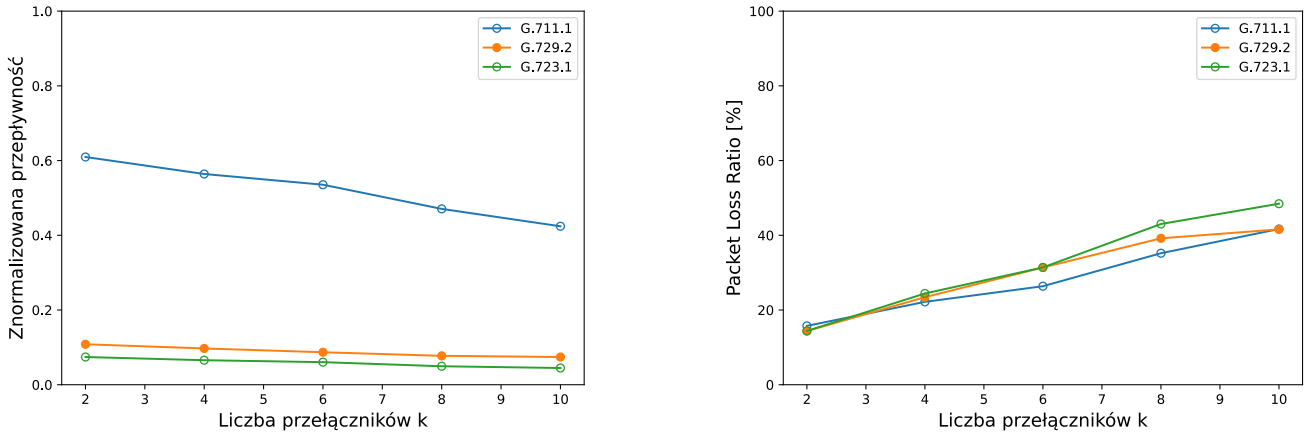
W sieci badano dwa scenariusze - z ustawionym opóźnieniem na łączu 50ms oraz z ustawionym, oprócz opóźnienia, *packet loss ratio* na poziomie 5%, parametrem bardzo niepożądanym w ruchu VoIP. Podsumowanie testowanych scenariuszy przedstawiono w Tabeli 17. Wyniki badań można zaobserwować na Rysunkach 18 oraz 19.

Scenariusz	Opóźnienie łącza [ms]	*Packet loss ratio* [%]	liczba hostów (n)
1	50	-	10
2	50	5	10

Tabela 17: Scenariusze symulacji ruchu VoIP w topologii *Linear*



Rysunek 18: Porównanie wyników dla scenariusza 1



(a) Przepływność

(b) Packet loss ratio

Rysunek 19: Porównanie wyników dla scenariusza 2

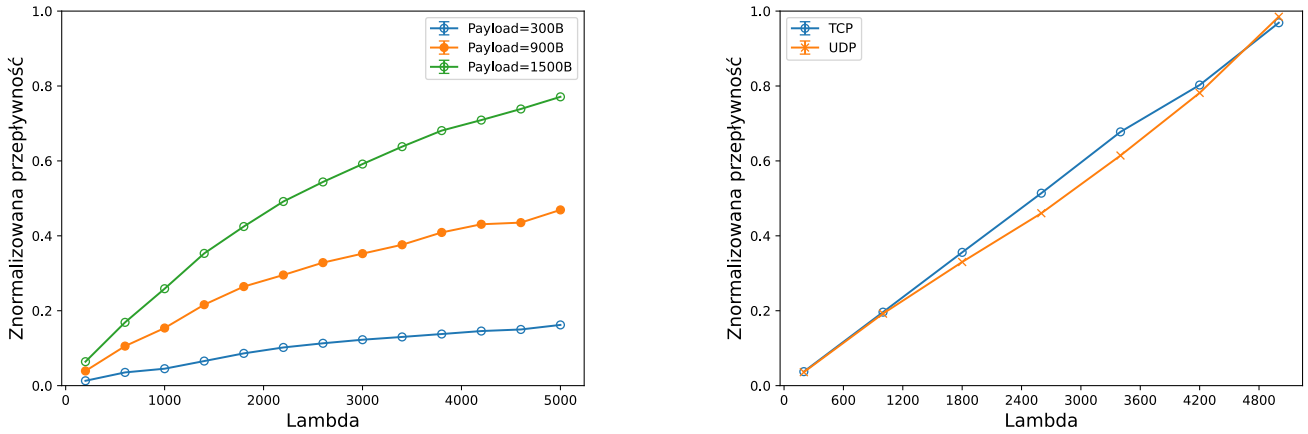
Dla ustawionego wyłącznie opóźnienia, przepływność w sieci nie zmieniała się, pomimo zwiększania liczby przełączników. Eksperyment powtórzono dla większego opóźnienia na łączu (200ms), lecz rezultaty były bardzo zbliżone. W przypadku scenariusza z ustawioną stratą pakietów, przepływność zmniejsza się w zależności od liczby przełączników w sieci. Warto także zauważyć, jak zmienia się procentowy udział utraconych pakietów - dla ruchu głosowego strata 30% pakietów oznacza niezrozumiałą i przerywaną rozmowę. Najmniejszą stratę pakietów gwarantuje użycie domyślnego kodeka G.711.1. Wraz ze zwiększającą się liczbą przełączników w sieci pogarszało się także działanie generatora - generowanie pakietów trwało coraz dłużej i coraz częściej nie mógł znaleźć trasy do celu. Z tego powodu skrócono eksperyment do pięciu prób.

5.2.2 Rozkłady prawdopodobieństwa

Opracowanie: Jan Ściga

Scenariusz	Rodzaj topologii	Rodzaj wykorzystanego rozkładu	Opóźnienie na łączu [ms]
1	Linear (n=10, k=2)	Poisson	-
2	Linear (n=20, k=4)	Wykładniczy	30

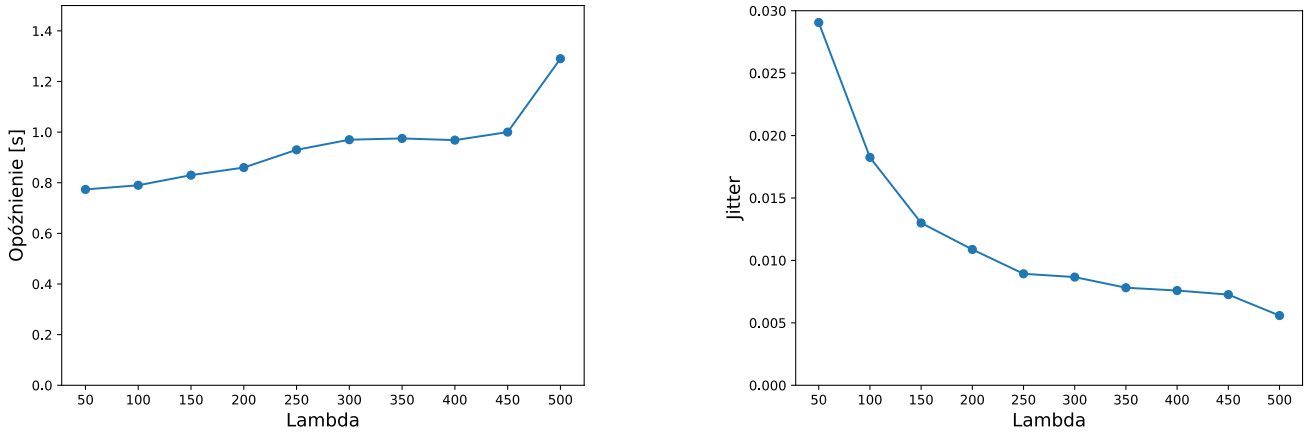
Tabela 20: Scenariusze symulacji rozkładów prawdopodobieństw



(a) Lambda jako intensywność generowania pakietów

(b) Lambda jako zmienność wielkości pakietu

Rysunek 21: Wyniki dla scenariusza 1



(a) Opóźnienie

(b) Jitter

Rysunek 22: Wyniki dla scenariusza 2

Przedstawienie konfiguracji: Symulacja rozkładów prawdopodobieństw została przeprowadzona w dwóch scenariuszach wykorzystujących odpowiednio rozkłady Poissona i wykładniczy dostępne w narzędziu D-ITG. Obydwie symulacje wykonano na bazie emulowanej wcześniej topologii *Linear*, która jednak w przypadku pierwszym zakłada mniejszą liczbę przełączników oraz hostów (10,2) niż w drugim (20,4). Dodatkowo, w drugim scenariuszu skonfigurowano opóźnienie na łączu (30 ms), podczas gdy w pierwszym scenariuszu pominięto ten parametr. Badanymi metrykami są: przepływność (w scenariuszu 1), a także jitter oraz opóźnienie (w scenariuszu 2).

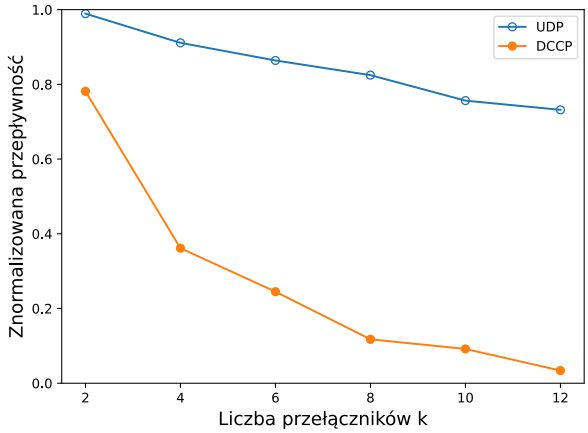
Omówienie wyników: W scenariuszu pierwszym widocznym na Rysunku 21 skupiono się na zbadaniu możliwości wykorzystania rozkładów do modelowania intensywności ruchu (21a) oraz zmienności wielkości pakietów (21b). W obu przypadkach przepływność wzrasta, co wynika odpowiednio z większego nasycenia sieci (21a) oraz zwiększenia efektywności wykorzystania kanału przez zmniejszenie narzutu sygnalizacyjnego (21b). W scenariuszu drugim widocznym na Rysunku 22 badany jest wpływ intensywności generowania pakietu na opóźnienie (22a) oraz jitter (22b). Wzrost opóźnienia spowodowany jest tym, że pakiety przychodzące do węzłów muszą być coraz dłużej kolejkowane poprzez narastającą wielkość ruchu. Zaobserwowano także, że przy zwiększającym się ruchu, jitter w coraz większym stopniu zależy od czasu kolejkowania, a nie czasu obsługi jak w przypadku ruchu o małej intensywności, co powoduje mniejsze zróżnicowanie opóźnień w scenariuszu 2.

5.2.3 Porównanie protokołów UDP oraz DCCP dla ruchu *bursty*

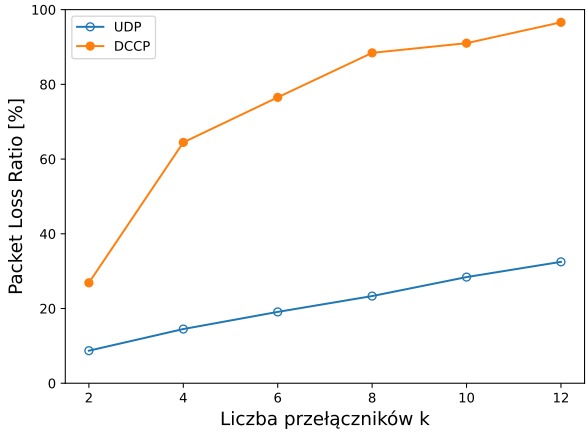
Opracowanie: Hubert Majdański

Scenariusz	Parametr na łączu	Wartość parametru
1	Strata Pakietów	5 %
2	Opóźnienie	30 ms

Tabela 23: Scenariusze symulacji porównania protokołów UDP i DCCP dla ruchu *bursty*

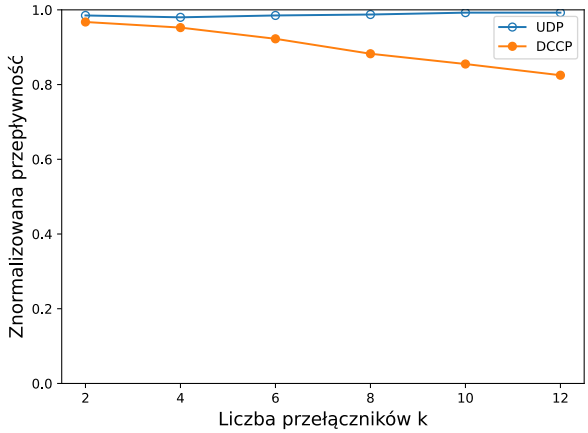


(a) Znormalizowana przepływność

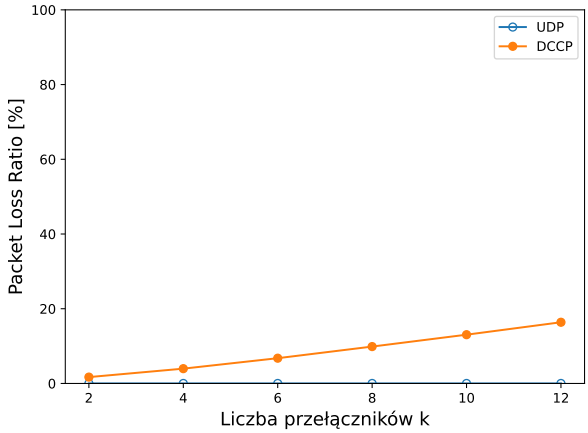


(b) Packet loss ratio

Rysunek 24: Wyniki dla scenariusza 1



(a) Znormalizowana przepływność



(b) Packet loss ratio

Rysunek 25: Wyniki dla scenariusza 2

Przedstawienie konfiguracji: Symulacja mająca na celu porównanie protokołów transportowych UDP oraz DCCP została przeprowadzona dla dwóch scenariuszy. Pakiety były generowane przy pomocy parametru *bursty* - pozwala on na zasymulowanie sieci, w której pakiety są generowane okresowo z pewnymi przerwami - czasy te mogą przyjmować wartość stałą lub losową. W scenariuszu pierwszym były one stałe (czas wysyłania pakietów oraz oczekiwania był podzielony po połowie). W przypadku drugim natomiast czas generacji był losowo wyznaczany przy pomocy rozkładu wykładniczego o średniej 100, a czas oczekiwania był generowany jako rozkład Weibulla o kształcie 10 i parametrze skali 100. Wyznaczonymi metrykami QoS są przepływność oraz *packet loss ratio* w przypadku obu scenariuszy.

Omówienie wyników: *Datagram Congestion Control Protocol* DCCP jest protokołem transportowym, zapewniającym mechanizmy kontroli przeciążenia. Zarówno UDP jak i DCCP są protokołami bezpołączeniowymi. DCCP zapewnia dodatkowo w porównaniu do protokołu UDP mechanizmy redukcji przepływu, a także mechanizm retransmisji w przypadku utraty pakietów. W wynikach jest to widoczne - w protokole DCCP poprzez mechanizm jego działania więcej pakietów jest traconych (a także wysyłanych) poprzez występowanie retransmisji. Dodatkowo wartą zauważenia rzeczą jest to, że protokół UDP posiadał dużo wyższą przepustowość znormalizowaną, dzięki czemu jest on wybierany w takich zastosowaniach jak m.in transmisja wideo, a protokół DCCP w zastosowaniach wymagających bezawaryjności.

6 Wnioski

Opracowanie: Justyna Gręda

W ramach projektu zbadano działanie generatora ruchu sieciowego D-ITG pod względem zapewnienia QoS w popularnych obecnie sieciach SDN. W tym celu użyty został emulator Mininet oraz kontroler POX. Podstawowe badania wykonano dla trzech predefiniowanych w wykorzystywanym emulatorze topologii: *Single*, *Linear* oraz *Tree*. Dla pojedynczego kontrolera i wielu hostów w topologii *Single* przepływność rośnie wraz ze wzrostem wielkości przesyłanego *payloadu*. Przy ustawionym wysokim *packet loss ratio* i protokole UDP, generator D-ITG sprawdza się lepiej niż jego prostszy odpowiednik iperf - pozwala osiągnąć większą część maksymalnej przepływności i zawsze znajduje drogę od hosta do celu. W topologii *Linear* sprawdzono wpływ liczby przełączników na przepływność i opóźnienie w sieci przy zadanym opóźnieniu łącza i wielkości *payloadu*. Wraz ze zwiększaniem się liczby przełączników przepływność maleje, a opóźnienie zwiększa się. Dla większego ustawionego opóźnienia łącza i większej liczby hostów, zmieniają one swoje wartości znacznie szybciej. Generator iperf pozwolił na osiągnięcie nieco wyższej przepływności i niższego opóźnienia niż D-ITG. Badając topologię *Tree* sprawdzono, że rozpiętość drzewa nie wpływa na przepływność oraz opóźnienie w sieci.

Przeanalizowano także bardziej zaawansowane funkcje generatora D-ITG. W pierwszej kolejności, zgodnie z tematem projektu sprawdzono, jak gwarantowana jest jakość obsługi dla charakterystycznych typów ruchu. Wybrano ruch generowany przez użytkowników popularnej gry *Counter Strike* oraz topologię *Linear*. Przy stałym opóźnieniu łącza opóźnienie w sieci rosło liniowo wraz z liczbą przełączników i nie zależało od tego, czy gracz aktualnie jest aktywny, czy nic nie robi. Gracz będący w trakcie meczu generował więcej ruchu, dzięki czemu przepływność była większa i niezależna od liczby przełączników. Sprawdzono także ruch VoIP dla trzech różnych kodeków audio. Gdy ustawiono wyłącznie opóźnienie na łączu, przepływność w sieci nie zmieniała się wraz ze wzrostem liczby przełączników. Jednak gdy dodano także *packet loss ratio*, zaczęła ona spadać. Najmniejszą procentową stratę pakietów gwarantował domyślny kodek G.711.1. Przeprowadzono również symulację rozkładów prawdopodobieństwa w topologii *Linear*. Dla rozkładu Poissona, wraz ze wzrostem parametru *Lambda* rośnie znormalizowana przepływność, zarówno dla różnych wielkości *payloadu*, jak i różnych protokołów sieciowych (TCP oraz UDP). Dla rozkładu wykładniczego, opóźnienie rośnie wraz ze wzrostem *Lambda*, ponieważ pakiety są coraz dłużej kolejkowane. Wykazano także, że jitter, który w scenariuszu znacznie spadał wraz ze wzrostem *Lambda* zależy od czasu kolejkowania pakietów. Oprócz tego, porównano działanie bezpołączeniowych protokołów UDP I DCCP dla ruchu typu *bursty*. Wykazano, że przy zwiększającej się liczbie przełączników w topologii *Linear* przepływność UDP jest wyższa i nie maleje, w przeciwieństwie do DCCP. UDP zapewnia także niższy *packet loss ratio*, przez co lepiej sprawdzi się w ruchu takim jak transmisja wideo.

Wykonane badania pozwoliły dogłębnie zapoznać się z generatorem D-ITG oraz porównać go z innymi odpowiednikami. Ponadto sprawdzono także gwarantowanie jakości obsługi w sieciach SDN, które dzięki swojej budowie stają się coraz bardziej popularne. Kluczowe jest więc przeprowadzanie w nich testów QoS pod wieloma względami, między innymi tymi przedstawionymi w projekcie.

Bibliografia

[1] Alessio Botta and Alberto Dainotti i Antonio Pescapè. "A tool for the generation of realistic network workload for emerging networking scenarios". W: *Computer Networks* 56.15 (2012), s. 3531–3547.

[2] Rashid Amin, Martin Reisslein i Nadir Shah. "Hybrid SDN Networks: A Survey of Existing Approaches". W: *IEEE Communications Surveys Tutorials* 20.4 (2018), s. 3259–3306. DOI: 10.1109/COMST.2018.2837161.

[3] Idris Z. Bholebawa i Upena D. Dalal. "Performance Analysis of SDN/OpenFlow Controllers: POX Versus Floodlight". W: *Wireless Personal Communications* 98.2 (sierp. 2017), s. 1679–1699. DOI: 10.1007/s11277-017-4939-z. URL: https://doi.org/10.1007/s11277-017-4939-z.

[4] Neelam Gupta i in. "A Comparative Study of Software Defined Networking Controllers Using Mininet". W: *Electronics* 11.17 (sierp. 2022), s. 2715. DOI: 10.3390/electronics11172715. URL: https://doi.org/10.3390/electronics11172715.

[5] Murat Karakus i Arjan Durresi. "Quality of Service (QoS) in Software Defined Networking (SDN): A survey". W: *Journal of Network and Computer Applications* 80 (2017), s. 200–218. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2016.12.019. URL: https://www.sciencedirect.com/science/article/pii/S1084804516303186.

[6] Surendra Kumar Keshari, Vineet Kansal i Sumit Kumar. "A Systematic Review of Quality of Services (QoS) in Software Defined Networking (SDN)". W: *Wireless Personal Communications* 116.3 (lut. 2021), s. 2593–2614. ISSN: 1572-834X. DOI: 10.1007/s11277-020-07812-2. URL: https://doi.org/10.1007/s11277-020-07812-2.

[7] Samad S. Kolahi i in. "Performance Monitoring of Various Network Traffic Generators". W: *2011 UkSim 13th International Conference on Computer Modelling and Simulation*. 2011, s. 501–506. DOI: 10.1109/UKSIM.2011.102.

[8] V. Krishna Nandivada i J. Palsberg. "Timing analysis of TCP servers for surviving denial-of-service attacks". W: *11th IEEE Real Time and Embedded Technology and Applications Symposium*. 2005, s. 541–549. DOI: 10.1109/RTAS.2005.54.

[9] Lusani Mamushiane i Themba Shozi. "A QoS-based Evaluation of SDN Controllers: ONOS and OpenDayLight". W: *2021 IST-Africa Conference (IST-Africa)*. 2021, s. 1–10.

[10] Aref Meddeb. "Internet QoS: Pieces of the puzzle". W: *IEEE Communications Magazine* 48.1 (2010), s. 86–94. DOI: 10.1109/MCOM.2010.5394035.

[11] Sudhakar Mishra, Shefali Pratap Sonavane i Anil Kumar Gupta. "Study of Traffic Generation Tools". W: 2015.

[12] Rogerio Leao Santos de Oliveira i in. "Using Mininet for emulation and prototyping Software-Defined Networks". W: *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*. IEEE, czer. 2014. DOI: 10.1109/colcomcon.2014.6860404. URL: https://doi.org/10.1109/colcomcon.2014.6860404.

[13] Ligia Rodrigues Prete i in. "Simulation in an SDN network scenario using the POX Controller". W: *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*. IEEE, czer. 2014. DOI: 10.1109/colcomcon.2014.6860403. URL: https://doi.org/10.1109/colcomcon.2014.6860403.

[14] Slavica Tomovic, Neeli Prasad i Igor Radusinovic. "SDN control framework for QoS provisioning". W: *2014 22nd Telecommunications Forum Telfor (TELFOR)*. 2014, s. 111–114. DOI: 10.1109/TELFOR.2014.7034369.