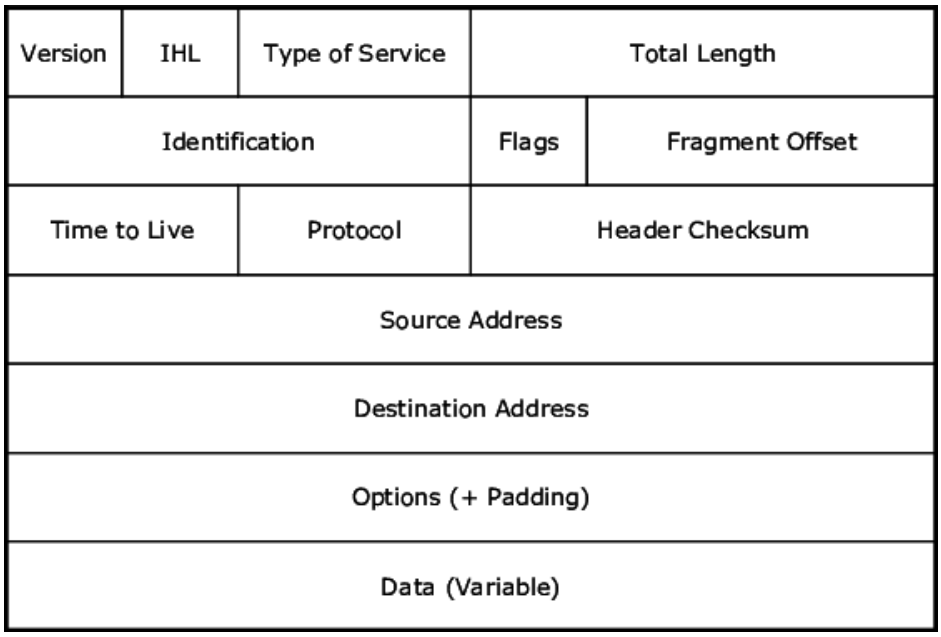
Implementacja QoS za pomocą P4 w Mininet

Niniejsze ćwiczenie zostało stworzone na podstawie oficjalnego tutoriala z repozytorium języka P4: https://github.com/p4lang/tutorials/tree/master/exercises/qos. Adaptacji tego tutoriala dokonali: Mikołaj Kardyś, Jakub Janczarski i Piotr Makarewicz.

Celem jest implementacja mechanizmu Quality of Service w warstwie trzeciej modelu ISO/OSI z wykorzystaniem języka programowania P4 i środowiska Mininet.

Wstęp teoretyczny

Quality of Service - zbiór mechanizmów pozwalających na priorytetyzację i rezerwację zasobów. Pozwalają one na nadanie oddzielnych priorytetów między innymi dla różnych aplikacji, użytkowników a także, jak w wypadku tego ćwiczenia, protokołów transferu danych.



W pakiecie IP pole ToS (Type of Service) składa się z 8 bitów:

- 6 starszych bitów - DSCP - klasa ruchu
- 2 młodsze bity - ECN - służą do informowania o obciążeniu łącza - w tym ćwiczeniu te bity nas nie interesują

Sieci mają możliwość rozróżniania do 64 klas ruchu, używając 64 możliwych wartości DSCP. Większość sieci wykorzystuje główne klasy związane z odpowiednimi zachowaniami:

- Default Forwarding (DF) — zwykle ruch typu best-effort
- Expedited Forwarding (EF) — ruch o małych stratach i opóźnieniach

- Assured Forwarding (AF) — ruch posiada gwarancję dostarczenia przy spełnionych określonych warunkach, zazwyczaj związanych z nie przekraczaniem ustalonej wielkości ruchu
- Voice Admit — ma tą samą charakterystykę co EF, natomiast używa również procedury Call Admission Control

Przygotowanie środowiska

1. Uruchom oficjalną maszynę wirtualną P4. To ćwiczenie było testowane na maszynie w wersji z czerwca 2023 r. Obraz można pobrać ze strony: https://github.com/jafingerhut/p4-guide/blob/master/bin/README-install-troubleshooting.md

2. Pobierz plik qos-demo.zip z miejsca wskazanego przez prowadzącego zajęcia.

3. Rozpakuj archiwum zip w folderze `~/tutorials/exercises`

Instrukcja wykonania ćwiczenia

1. Przyglądnij się plikowi `topology.json`. Z jakich elementów składa się topologia sieci w tym ćwiczeniu i jak są ze sobą połączone?

2. Uruchom aplikację w początkowej wersji za pomocą komendy:

```
$ make
```

   Jeśli kompilacja przebiegnie poprawnie, powinna uruchomić się konsola mininet. W razie, gdyby coś po drodze poszło nie tak, wyczyść środowisko komendą:

```
$ make clean
```

3. W konsoli mininet uruchom terminale dla hostów h1 i h11, aby wykonać test komunikacji między nimi:

```
mininet>  xterm h1 h11
```

4. W terminalu hosta h11 uruchom skrypt, który wyświetla wszystkie pakiety, które przychodzą do hosta h11:

```
$ ./receive.py
```

W terminalu hosta h1:

```
$ ./send.py --p=TCP --des=10.0.1.11 --m="Hello world" --dur=3

(...)

$ ./send.py --p=UDP --des=10.0.1.11 --m="Hello world" --dur=3
```

5. Przyglądnij się informacjom wyświetlanym wewnątrz terminala hosta h11. Czy komunikacja przebiega pomyślnie? Jakie wartości przyjmuje pole `ipv4.tos` dla każdego z protokołów? Do której klasy DSCP należą odbierane pakiety?

6. Uzupełnij sekcje oznaczone przez TODO wewnątrz akcji `default_forwarding`, `expedited_forwarding` oraz `voice_admit`
   *Wskazówka: wartości których należy użyć są ściśle określone:
   https://www.iana.org/assignments/dscp-registry/dscp-registry.xhtml

7. Uzupełnij sekcje oznaczone przez TODO wewnątrz sekcji `apply` elementu `MyIngress`. Naszym celem jest to, aby switch przydzielał wszystkim pakietom UDP klasę “Expedited Forwarding”, a pakietom TCP klasę “Voice Admit”.

8. Ponownie przetestuj zachowanie przykładu, powtarzając kroki 2. oraz 3.

9. Czy wartości, które przyjmuje tym razem pole `ipv4.tos` są inne niż w punkcie 5.? Jakie są dokładnie i z czego to wynika?