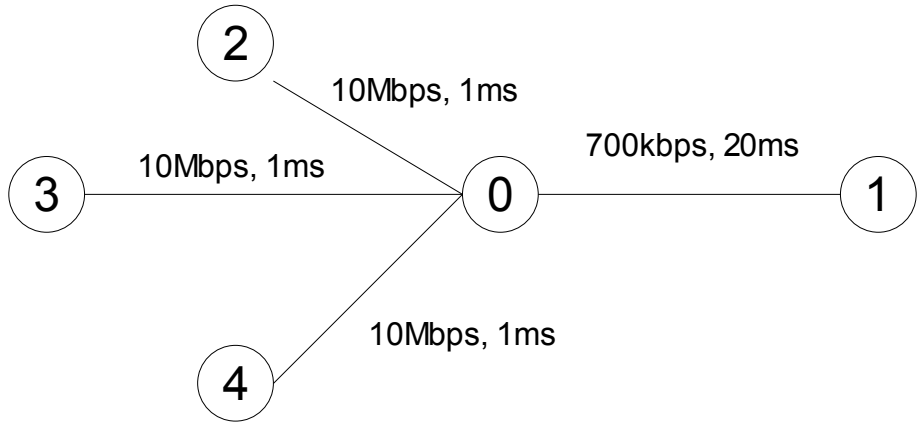
Opracował: Konrad Kawecki <cogi@pjwstk.edu.pl>
Krzysztof Rządca <krz@pjwstk.edu.pl>

# Tematyka

Symulacje z wykorzystaniem protokołu TCP oraz zdarzenia losowe, kontynuacja.

# TCP, kontynuacja

Mamy sieć jak na ilustracji 1.




*Ilustracja 1: Schemat sieci*

***Ćwiczenie 1***

Stwórz topologię jak na ilustracji 1. Połączenia między węzłami z kolejką typu DropTail.

Po sprawdzeniu topologii wprowadzimy połączenia FTP.

***Ćwiczenie 2***

Węzły od 2 do 4 nawiązują połączenie FTP z "serwerem" - węzłem 1. Utwórz te połączenia. Dla każdego z połączeń dodaj oddzielnego odbiorcę TCP. Pokoloruj przepływy. Włącz podgląd kolejki wychodzącej w węźle 1.

***Ćwiczenie 3***

Zaobserwuj, czy w kolejce pakietów oczekujących na przesłanie łączem 0-1 pakiety pochodzące od różnych nadawców są ze sobą wymieszane, czy raczej występują w blokach. Czym wytłumaczysz takie zachowanie?

Sprawdź ile danych zostało wysłanych/odebranych przez każde z połączeń FTP. Przez które połączenie zostało wysłanych najwięcej danych? Dlaczego akurat przez te?

Na wspólnym wykresie przedstaw wielkość okna każdego z połączeń TCP. Na wykresie zaobserwuj fazy wzrostu eksponencjalnego (slow start) i liniowego (congestion avoidance). Dlaczego wykresy różnią się? Czym wytłumaczysz zresetowanie wielkości okna u niektórych nadawców? Zaobserwuj zmiany parametru `ssthresh_` określającego moment przejścia pomiędzy slow start i congestion avoidance. Porównaj zmiany `ssthresh_` ze zmianami wielkości okna.

Na łączu pomiędzy węzłami 0 a 1 zmień kolejkę na SFQ. Powtórz poprzednie doświadczenia. Czy kolejka SFQ bardziej "sprawiedliwie" współdzieli łącze?

Popatrz na wykres wielkości okien. Czy w dłuższym okresie czasu TCP jest w stanie poprawnie "wyczuć" przepustowość dostępną na łączu? Czy teoretyczna przepustowość łącza została w pełni wykorzystana przez przepływy? W których momentach TCP działa niewydajnie? Zasugeruj "idealny" algorytm w pełni wykorzystujący dostępną przepustowość.

Spróbuj nieco rozsynchronizować czasy startów komunikacji. Czy ilość przesłanych danych wzrosła? (Uwaga: aby porównanie było sprawiedliwe, musisz liczyć dane przesłane tylko wtedy, gdy wszyscy trzej nadawcy działali jednocześnie)

Stop-and-wait to bardzo prosty protokół, w którym nadawca wysyła jedną ramkę, czeka na jej potwierdzenie, a następnie wysyła kolejną ramkę. Możemy zasymulować działanie tego protokołu na TCP ustawiając bieżącą i maksymalną wielkość okna nadawcy na 1:

```
$tcp_sender1 set window_ 1
$tcp_sender1 set maxcwnd_ 1
```

***Ćwiczenie 4***

Uruchom ponownie symulację z poprzedniego ćwiczenia i zapisz, ile danych przesłał każdy z nadawców. Następnie zmodyfikuj nadawców tak, by wszyscy używali protokołu stop-and-wait. Uruchom symulację i porównaj ilości przesłanych danych. Policz całkowitą przepustowość.

Który z parametrów łącza - przepustowość czy opóźnienie – decyduje o tym, że stop-and-wait jest niewydajne? W czym TCP ulepsza stop-and-wait?

# Losowość, kontynuacja

Nauczymy się w jaki sposób możemy losowo zaczynać i kończyć transmisję. Wykorzystamy do tego celu generator liczb losowych.

```
set rng [new RNG]
$rng seed 7

set RVstart [new RandomVariable/Uniform]
$RVstart set min_ 0
$RVstart set max_ 10
$RVstart use-rng $rng
```

Powyższy fragment skryptu inicjalizuje generator liczb losowych. Losową wartość odczytujemy w następujący sposób.

```
$RVstart value
```

***Ćwiczenie 5***

Do poprzedniego ćwiczenia dodaj losowy start i zakończenie transmisji. W pliku z danymi symulacji sprawdź dokładne czasy uruchomienia i zakończenia połączeń.