Opracował: Konrad Kawecki <cogi@pjwstk.edu.pl>
na podstawie materiałów: http://www.isi.edu/nsnam/ns/tutorial/index.html

Tematyka

Na ćwiczeniach zapoznamy się z symulatorem `ns-2` oraz narzędziem do wizualizacji symulacji `nam`. Napiszemy własne skrypty w języku `tcl` opisujące modele sieci komputerowych.

Podstawy NS-2, definiowanie węzłów

Symulacje opisujemy za pomocą skryptów w języku *tcl*. Na początku definiujemy obiekt symulatora.

```
set ns [new Simulator]
```

Następnie tworzymy plik w którym znajdą się wynikowe dane potrzebne do wizualizacji symulacji (dane dla programu `nam`).

```
set nf [open out.nam w]
$ns namtrace-all $nf
```

W pierwszej linii otwieramy plik (tylko do zapisu) `out.nam` i przypisujemy go do uchwytu `nf`. W drugiej linii informujemy obiekt symulatora aby wszystkie dane niezbędne do wizualizacji przesyłał do tego pliku.

Następnym krokiem jest napisanie procedury kończącej symulację. W procedurze tej zamykamy otwarte pliki i uruchamiamy program `nam`.

```
proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0
}
```

Kolejnym krokiem jest zdefiniowanie węzłów oraz połączenia między węzłami. Zaczniemy od definiowania węzłów.

```
set n0 [$ns node]
set n1 [$ns node]
```

Za pomocą polecenia `$ns node` tworzymy nowy węzeł. Węzły przypisujemy do uchwytów `n0` i `n1`.

Poniższa linia definiuje dwustronne połączenie między węzłami. Połączenie ma przepustowość 1Mbit/s, opóźnienie 10ms oraz kolejkę typu droptail (pakiety przychodzące, które nie mieszczą się w kolejce, są odrzucane).

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

Następnie umieszczamy informacje o zakończeniu symulacji (symulację kończymy po 5 sekundach).

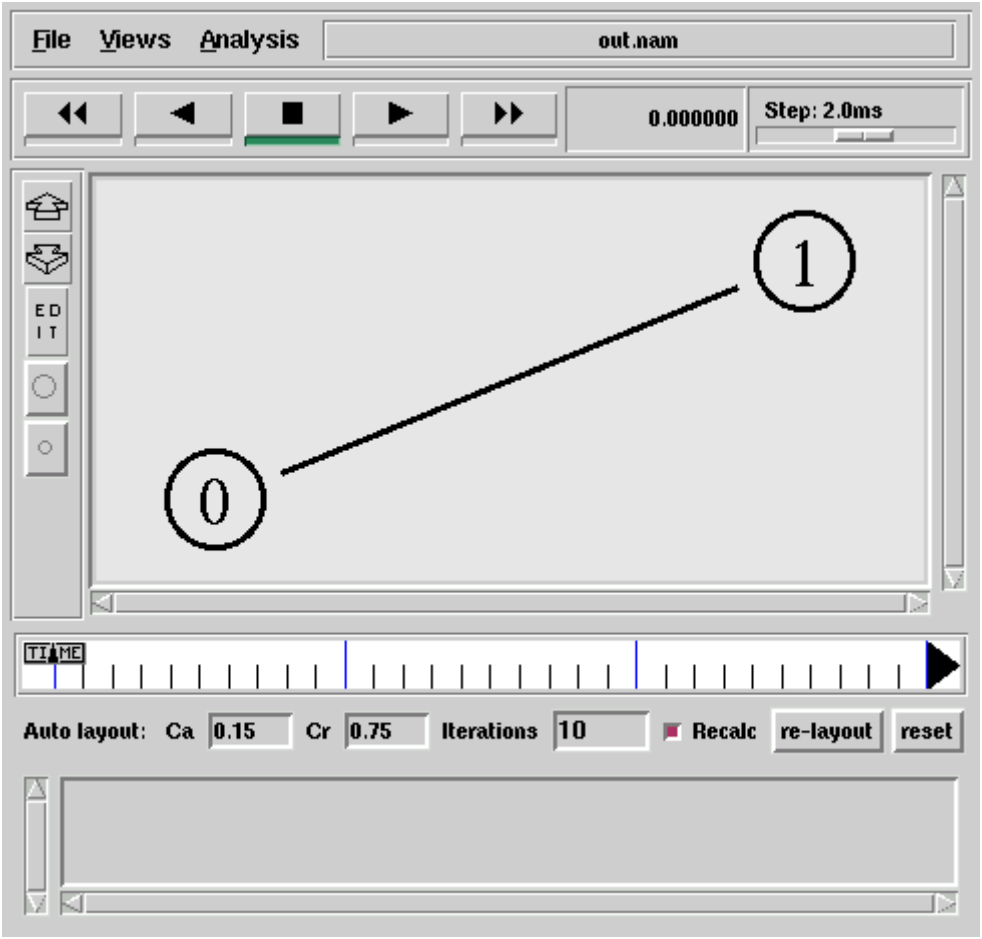
```
$ns at 5.0 "finish"
```

Ostatnia linia w naszym pierwszym skrypcie uruchamia symulację.

```
$ns run
```

Zakładając, że nasz skrypt zapisaliśmy w pliku o nazwie `symulacja1.tcl`, symulator `ns-2` uruchamiamy wykonując poniższe polecenie.

```
ns symulacja1.tcl
```



Ilustracja 1: Wynik uruchomienia naszej pierwszej symulacji.

Na ilustracji 1 widoczny jest efekt uruchomienia naszego pierwszego skryptu.

Ćwiczenie 1

Utwórz plik z powyższą symulacją i sprawdź jego działanie

Strumienie danych

W naszej symulacji nie dzieje się zbyt wiele. Czas na zdefiniowanie strumieni danych. Do węzłów przypiszemy agentów wysyłających/odbierających informacje. Agent w węźle `n0`

będzie wysyłał informacje a agent w węźle `n1` będzie je odbierał.

```
#Tworzymy agenta UDP i przypisujemy go do węzła n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

#Tworzymy źródło danych CBR i przypisujemy je do agenta $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

Powyższy fragment skryptu tworzy agenta UDP `udp0` i przypisuje go do węzła `n0`. Następnie tworzymy źródło danych typu Constant Bit Rate (CBR). Ustawiamy wielkość pakietu na 500 bajtów a czas pomiędzy transmisją pakietów wynosi 0.005 sekundy (wysyłamy 200 pakietów na sekundę).

Do węzła `n1` przypisujemy agenta odbiorce.

```
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
```

Aby przeprowadzić transmisję musimy połączyć agentów: nadawcę i odbiorcę.

```
$ns connect $udp0 $null0
```

Następnie informujemy agenta wysyłającego dane o tym kiedy ma rozpocząć transmisję.

```
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

Ćwiczenie 2

Po wprowadzeniu powyższych zmian uruchom symulację. Poeksperymentuj z parametrami źródła danych CBR oraz z interfejsem programu `nam`. Sprawdź czy można uzyskać informacje o pojedynczych pakietach. Sprawdź status łącza.

Samodzielnie dodaj kolejny węzeł i kolejne źródło danych. Określ odpowiednie połączenia. Wykonaj symulację.

Dodajemy ruter

W tej sekcji zajmiemy się topologią z czterema węzłami. Jeden węzeł będzie pełnił funkcję rutera przekazując informacje wysyłane przez dwa węzły do węzła czwartego.

Na początku definiujemy cztery węzły.

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
```

Zestawiamy połączenia między węzłami.

```
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms DropTail
```

Ćwiczenie 3

Uruchom powyższą symulację. Sprawdź jak wygląda rozłożenie węzłów. Sprawdź działanie funkcji „re-layout”.

Poniższy fragment skryptu kontroluje rozłożenie węzłów.

```
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
```

Ćwiczenie 4

Uruchom powyższą symulację. Sprawdź jak wygląda rozłożenie węzłów.

Następnie dodamy dwa źródła danych typu CBR i przypisujemy je do węzłów `n0` i `n1`. Do węzła `n3` przypisujemy agenta odbiorcę.

```
#Tworzymy agenta UDP i dodajemy go do węzła n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

#Tworzymy źródło danych CBR i przypisujemy je do agenta udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Tworzymy agenta UDP i dodajemy go do węzła n1
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

#Tworzymy źródło danych CBR i przypisujemy je do agenta udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
```

Tworzymy połączenia pomiędzy agentami.

```
$ns connect $udp0 $null0
$ns connect $udp1 $null0
```

Chcemy aby pierwszy agent CBR zaczął nadawać w pół sekundy po uruchomieniu symulacji i skończył nadawanie cztery sekundy później. Drugi agent CBR ma zacząć nadawanie w pierwszej sekundzie i skończyć po trzech sekundach.

```
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
```

Ćwiczenie 5

Uruchom powyższą symulację. Zwróć uwagę na straty pakietów.

Możemy zauważyć, że ruch na łączach `n0` do `n2`, `n1` do `n2` jest większy od przepustowości łącza `n2` do `n3`. Na pierwszy rzut oka nie możemy stwierdzić, które pakiety są odrzucane. Strumienie danych są pokazane na czarno.

Ćwiczenie 6

Oblicz ruch generowany na łączach `n0` do `n2`, `n1` do `n2`. Czy ruch generowany na tych łączach rzeczywiście przekracza możliwości łącza `n2-n3`?

Kolorowanie strumieni danych

Dodajmy poniższe dwie linie do definicji agentów.

```
$udp0 set class_ 1
$udp1 set class_ 2
```

Kolejny fragment programu dodajmy pod definicją obiektu symulatora.

```
$ns color 1 Blue
$ns color 2 Red
```

Dzięki tym zmianom możemy rozróżnić strumienie danych.

Ćwiczenie 6

Uruchom symulacje, zaobserwuj zmienione kolory strumieni danych.

Monitorowanie kolejki

Aby monitorować kolejkę pomiędzy węzłami `n2` i `n3` dodajmy poniższą linię

```
$ns duplex-link-op $n2 $n3 queuePos 0.5
```

Ćwiczenie 7

Uruchom symulację. Zaobserwuj straty pakietów.

Kolejka typu droptail nie jest sprawiedliwa. Zmienimy rodzaj kolejki tak, by łącze `n2-n3` było dzielone w sposób sprawiedliwy. Poniższy fragment ustawia kolejkę typu SFQ.

```
$ns duplex-link $n3 $n2 1Mb 10ms SFQ
```

Ćwiczenie 8

Uruchom symulację. Zaobserwuj straty pakietów. Czy kolejka SFQ różni się od kolejki typu droptail? Czy możemy zauważyć to na symulacji?