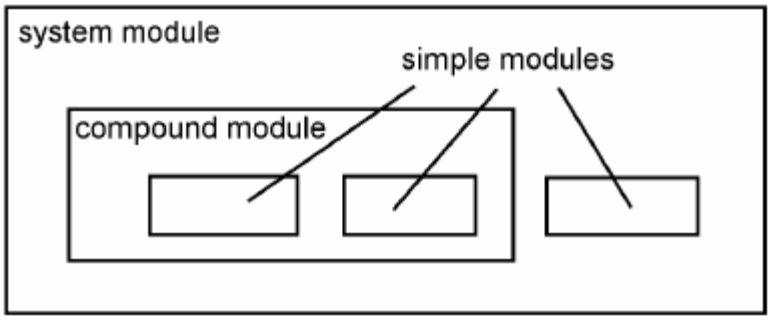
Lekcja 1. Środowisko OMNeT++

W przygotowanym materiale zostanie omówione środowisko OMNeT++, pokazane zostanie, w jaki sposób tworzyć modele topologii.

Język topologii NED

Model OMNeT ++ składa się z hierarchicznie zagnieżdżonych modułów. Moduły są opisane w języku NED i często odnoszą się do sieci (`networks`). Moduły komunikują się ze sobą przez przekazywane komunikaty (`messages`), które mogą zawierać dowolne struktury danych. Moduły mogą wysłać komunikaty bezpośrednio do ich celu lub wzdłuż pierwotnie zdefiniowanej ścieżki, przez bramy (`gates`) i połączenia (`connections`). Moduły mogą mieć własne parametry.

Moduł najwyższego poziomu to moduł systemowy (`system module`), który zawiera podmoduły (`submodules`), które same również mogą zawierać podmoduły.
Moduły w najniższym poziomie hierarchii są określone jako moduły proste (`simple module`) i ich zachowanie jest opisywane w języku C++ z wykorzystaniem bibliotek symulacyjnych. Moduły zawierające moduły proste to moduły złożone (`compound module`).



Rys. 1. Modularna struktura obiektu symulacji

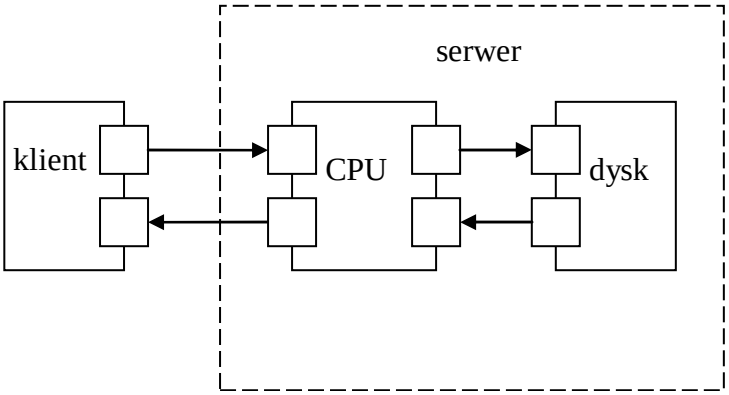
Programy mogą być uruchamiane z wykorzystaniem interfejsu linii poleceń lub dostarczanego środowiska graficznego z użyciem bibliotek Tcl/Tk (CMDEnv, TkEnv, Envir).

Do stworzenia symulacji potrzebne są pliki: *.ned, *.cpp, *.h oraz plik sterujący *omnetpp.ini.*
W prezentowanym materiale opisana zostanie budowa pliku .ned (informacje o konstrukcji plików *.h , *.cpp i omnetpp.ini znajdują się w *lekcji 2*).

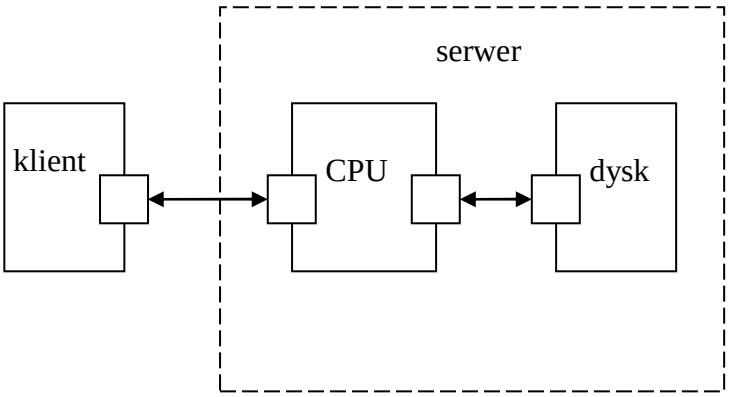
Przykład modelu

W celu zapoznania się z językiem NED zostanie wykonany przyjęty model serwera bazodanowego.

Schemat modelu przedstawiony jest na rysunku 2.



Rys. 2a. Schemat sieci do serwera bazodanowego
źródło: wykonanie własne



Rys. 2b. Schemat sieci do serwera bazodanowego
źródło: wykonanie własne

Model możemy tworzyć na dwa sposoby: definiując model w pliku tekstowym zapisanym z rozszerzeniem .ned lub tworząc poszczególne moduły w środowisku symulacyjnym. W celu dokładnego wyjaśnienia budowy modelu wykorzystany zostanie pierwszy sposób.
Pierwszym krokiem w tworzeniu modelu jest stworzenie roboczego katalogu o nazwie np. *Lekcja1*. Następnie należy stworzyć plik topologii, który jest plikiem tekstowym zapisanym z rozszerzeniem *.ned*.
Definiowanie modelu rozpoczyna się od modułów podrzędnych (`simple module`).
Wszystkie moduły są obiektami o typach zdefiniowanych przez użytkownika i zbudowane są z sekcji:

- `parameters` (parametry) – określa cechy topologii modelu lub modułu. Parametry mogą być wartościami o różnych typach (`string`, `bool`, `numeric`, `const`, `random`, `xml`). Umownie nazwy parametrów piszemy z małej litery.
- `gates` (bramy) – określa wejście – „`input`”, wyjście – „`output`” lub brama dwukierunkowa – „inout” modułu, przez które przesyłane są komunikaty. Każde połączenie modułów, które ma wejście musi mieć też wyjście. Do jednej bramy wyjściowej może być podłączona tylko jedna brama wejściowa i na odwrót – do każdej bramy wejściowej tylko jedna brama wyjściowa. Można również definiować wektory bram zwierające pewną liczbę pojedynczych bram za pomocą pary nawiasów [], a ich wielkość podać w module złożonym (więcej na ten temat znajduje się w Lekcji 4).

Dodatkowo moduły złożone mają sekcje:

- `connections` (połączenia) – określa połączenia między wejściami i wyjściami modułów podrzędnych, oraz połączenia modułów nadrzędnych z podrzędnymi.
- `submodules` (podmoduły) – zawiera definicje podmodułów. Typy podmodułów muszą być podane do kompilatora NED, dlatego to muszą one ukazać się wcześniej w tym

samym pliku NED albo zostać importowane z innego pliku NED. Można również definiować wektory podmodułów (więcej na ten temat w Lekcji 4).

Do zadania potrzebne są trzy podmoduły: dla klienta, procesora oraz dysku.
Umownie nazwy modułów zarówno prostych jak i złożonych zaczynają się z wielkich liter, więc nazwiemy je odpowiednio *Klient*, *Procesor*, *Dysk_twardy.*

```
simple Klient
{
        // parameters:
        //…
    gates:
        inout oddo_klienta;
}

simple Procesor
{
        // parameters:
        //…
    gates:
        inout oddo_klienta;
        inout oddo_dysku;
}

simple Dysk_twardy
{
        // parameters:
        //…
    gates:
        inout oddo_procesora;
}
```

Komentarze w plikach **.ned* umieszcza się po znaku dwóch ukośników slash (//). Komentarze są ignorowane przez kompilator NED.

Jak na razie nie potrzebujemy parametrów więc zostawimy pustą sekcje `parameters`.

Parametry są zmiennymi należącymi wyłącznie do danego modułu. Parametry są identyfikowane nazwą umownie rozpoczynającą się z małych liter.
Jeśli typ parametru jest pominięty to standardowo przyjęty jest typ `numeric`. Wartości parametrów mogą być podane w sekcji `parameters` lub w pliku konfiguracyjnym *omnetpp.ini* (więcej informacji na ten temat w lekcji nr.2). Jeśli nie są zadeklarowane wartości to symulator zapyta o nie przy starcie symulacji.

Tak jak już wcześniej było wspomniane połączenia bram nie mogą być multipleksowane, tzn. do jednego wejścia może być podłączone tylko jedno wyjście. Można zadeklarować większą liczbę bram w postaci wektora, używając w tym celu podwójnego kwadratowego nawiasu [].

Teraz możemy rozpocząć tworzenie modułów złożonych z wcześniej opisanych podmodułów.

Sekcje gates i parameters definiujemy analogicznie jak dla modułów prostych.

```
 module Serwer
{
        // parameters:
        //…
    gates:
        inout oddo_komputera;
    submodules:
        cpu: Procesor {
            // parameters:
            //…
        }
        dysk: Dysk_twardy {
            // parameters:
            //…
        }
    connections:
        oddo_komputera <--> cpu.oddo_klienta;
        cpu.oddo_dysku <--> dysk.oddo_procesora;
}
```

Podobnie jak wcześniej w modułach prostych tak i teraz w sekcji parameters nie zadeklarowaliśmy na razie żadnych zmiennych.

Parametry w tej sekcji mogą być wykorzystywane przez podmoduły. Mogą również zostać użyte w definiowaniu wewnętrznej struktury modułu złożonego, np. liczby submodułów. Parametry oddziałujące na wewnętrzną strukturę powinny zawsze być deklarowane jako const, wówczas za każdym razem jak będzie uzyskiwany do nich dostęp będą one miały taką samą wartość.

W sekcji submodules zadeklarowane są moduły podrzędne zadeklarowane wcześniej. W tej sekcji mogą znajdować się również przypisania konkretnych wartości do parametrów modułów podrzędnych. Jeśli w podmodułach zadeklarowane były wektory bramy to można w tej sekcji zadeklarować sekcję podrzędną gatesize, w której zostaną przypisane wielkości dla poszczególnych wektorów wejść/wyjść dla każdego z modułów prostych. Rozmiar bram nie jest obowiązkowy. Jeśli nie zostanie podany to zostanie stworzony z rozmiarem zerowym.

W sekcji Connections zadeklarowane są połączenia pomiędzy modułami w danym module złożonym.

Po zadeklarowaniu wszystkich modułów prostych i złożonych, do stworzenia kompletnego modelu potrzebna jest definicja sieci.

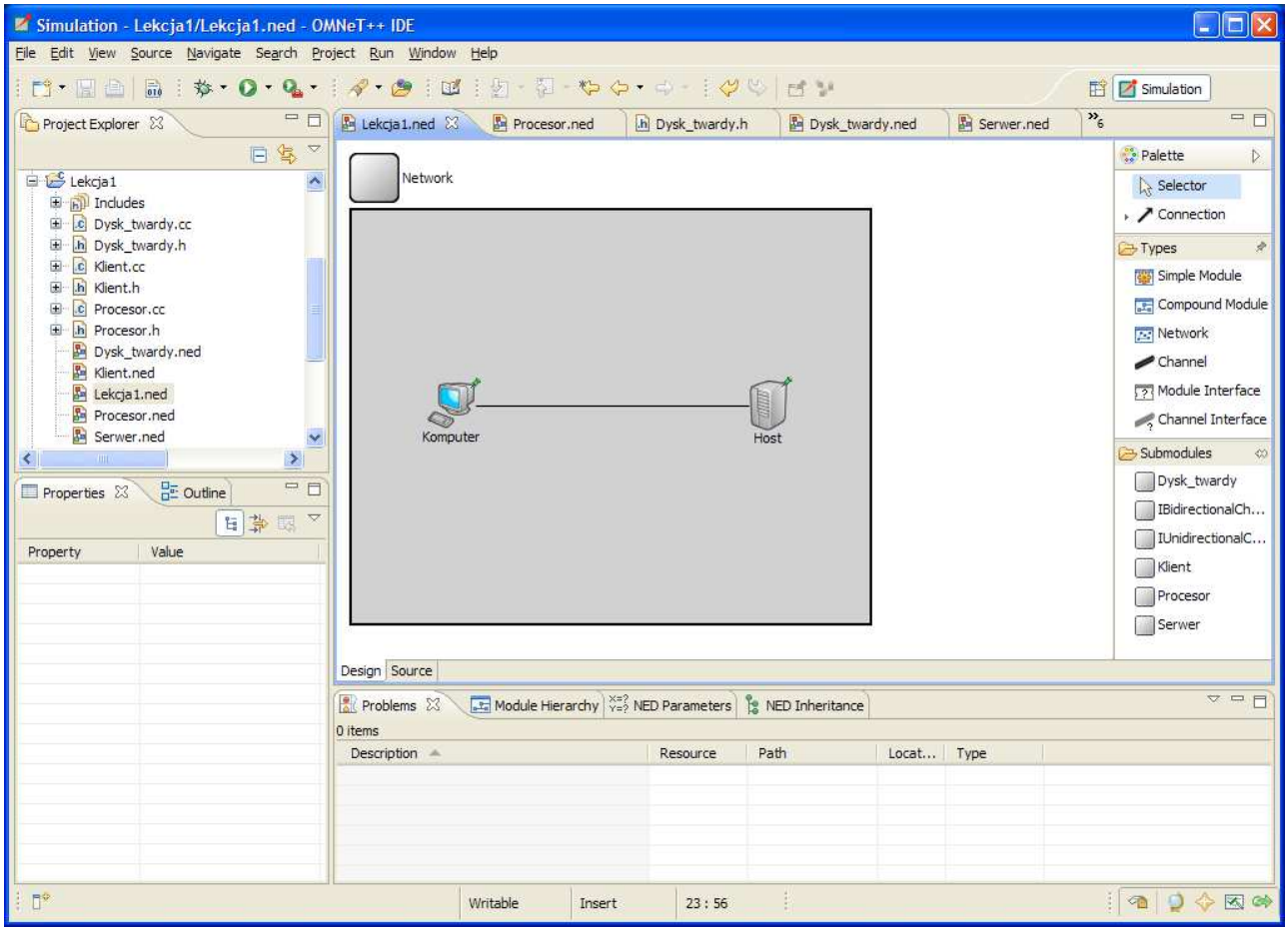
```
network Lekcja1
{
    @display("bgb=414,330");
    submodules:
        Host: Klient {
            @display("p=333,156;i=device/server2");
        }
        Komputer: Serwer {
            @display("p=79,156;i=device/pc4");
        }
    connections:
        Host.oddo_klienta <--> Komputer.oddo_komputera;
}
```

Sieć również może zawierać sekcję parameters.

Model topologii NED jest już gotowy do uruchomienia.

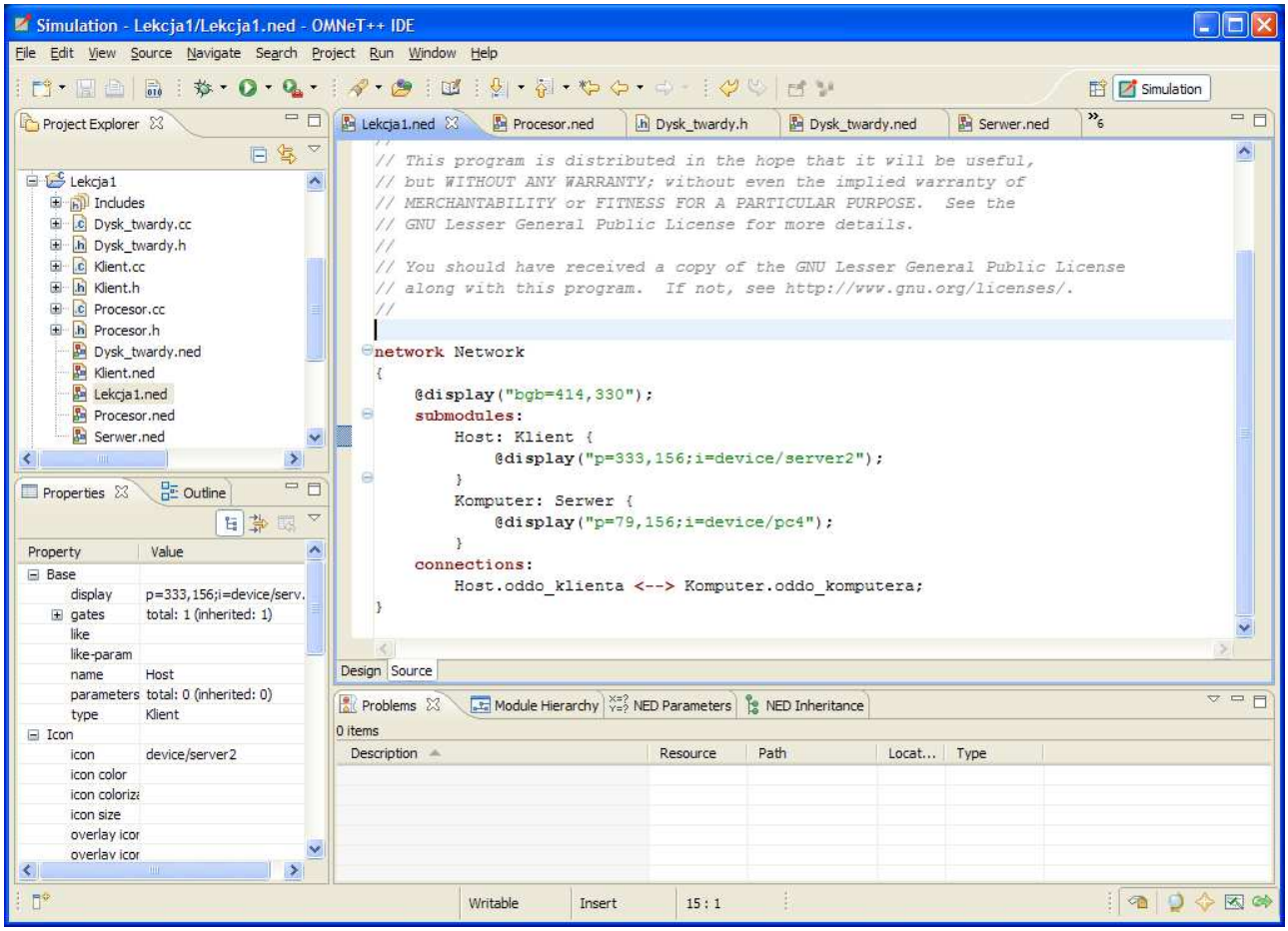
Edytor graficzny GNED.

Do budowy, edycji i odczytu plików .ned służy edytor graficzny.



Rys. 3. Model w trybie graficznym

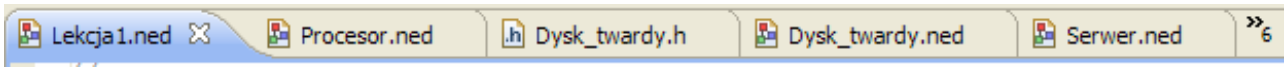
Przełączając miedzy zakładkami można oglądać model w trybie graficznym i w języku NED.



Rys. 4. Model w języku NED

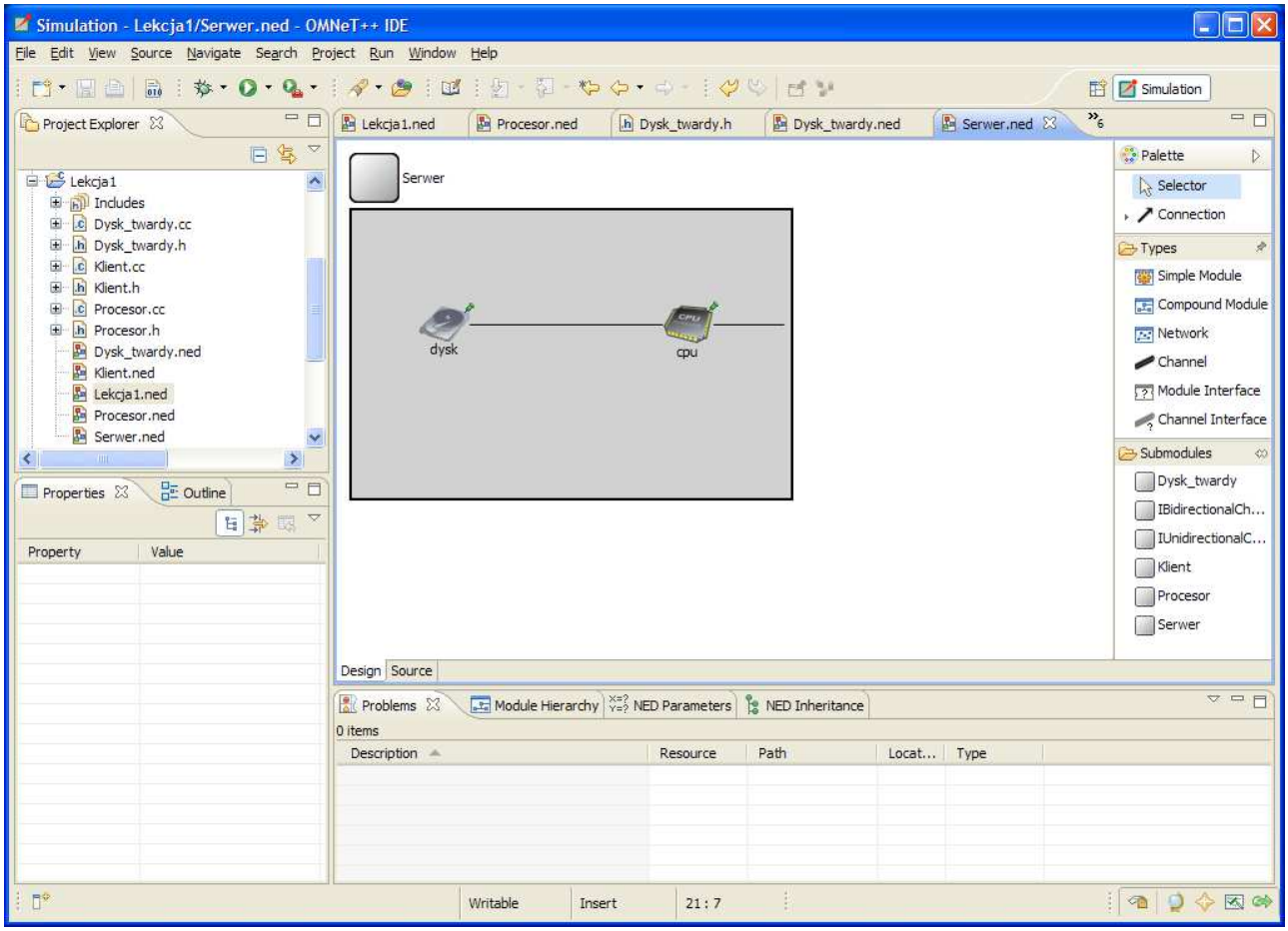
W lewej części okna aplikacji przedstawione jest drzewo modułów wraz z ich bramami, parametrami, a w przypadku modułów złożonych i sieci również submoduły i połączenia.

Jest możliwość wygodnego przełączania podglądu pomiędzy komponentami składowymi modelu (na górze okna aplikacji).



Rys. 5. Moduły

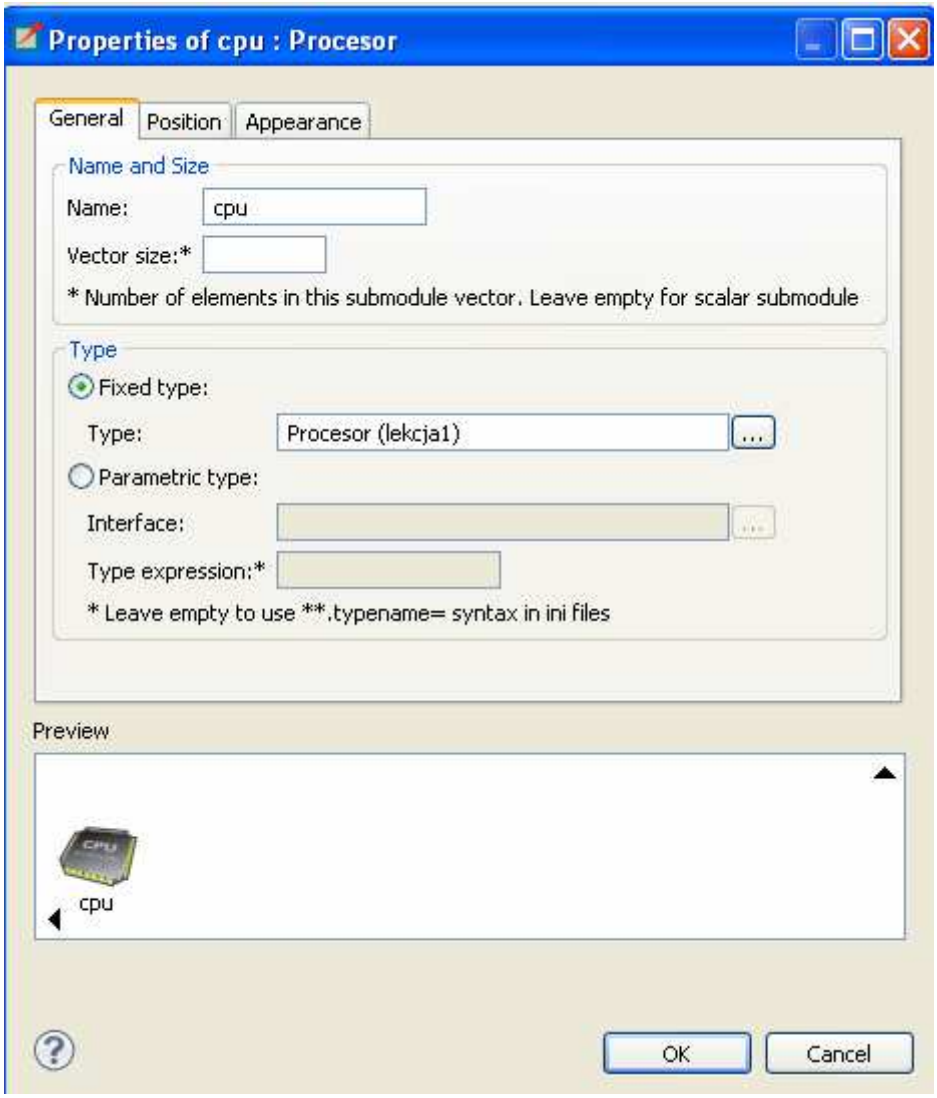
Można również otworzyć moduł Komputer w osobnym oknie klikając na ikonę modułu Serwer. Jest to przydatne przy obserwowaniu ruchu podczas działania symulacji.



Rys. 6. Model serwera w trybie graficznym

Automatycznie przy uruchomieniu modelu do kodu dopisane zostały linijki kodu: `display: „b=36,33;p=72,88”;` oraz `display: „b=30,40;p=281,88”;` odpowiadające ikonom modułu Klient i Serwer – ich rozmiar oraz położenie.

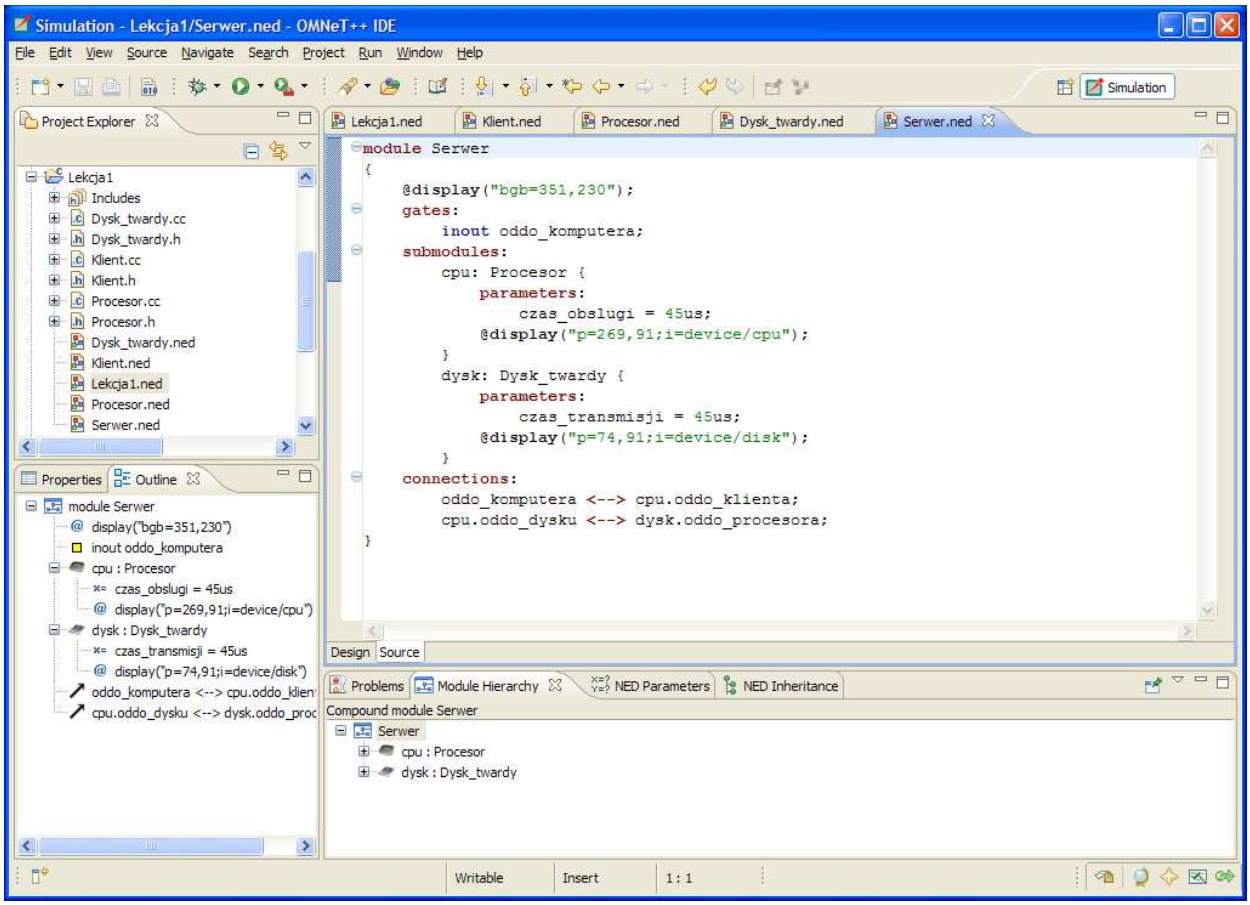
Klikają prawym przyciskiem myszki na poszczególne elementy modelu można oglądać i zmieniać parametry i właściwości podmodułów.



Rys. 7. Okno ustawień submodułu

W zakładce *General* znajdują się pola nazwy modułu, typ modułu i rozmiar wektora. W zakładce Position można zmienić pozycję modułu, a zakładka Appearance pozwala na zmianę jego wyglądu.

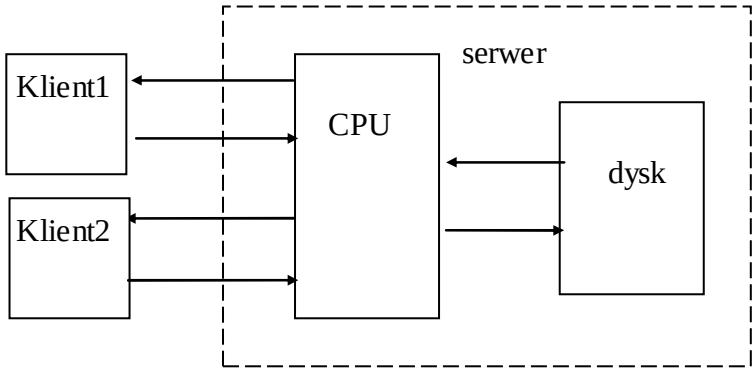
Można także zmienić położenie i wygląd modułu bezpośrednio w pliku *.ned.



Rys. 8. Moduł komputer w języku NED

Zadanie do samodzielnego przygotowania:

Wykonaj model topologii NED dla sieci przedstawionej na schemacie.



Rys. 10. Schemat serwera bazodanowego