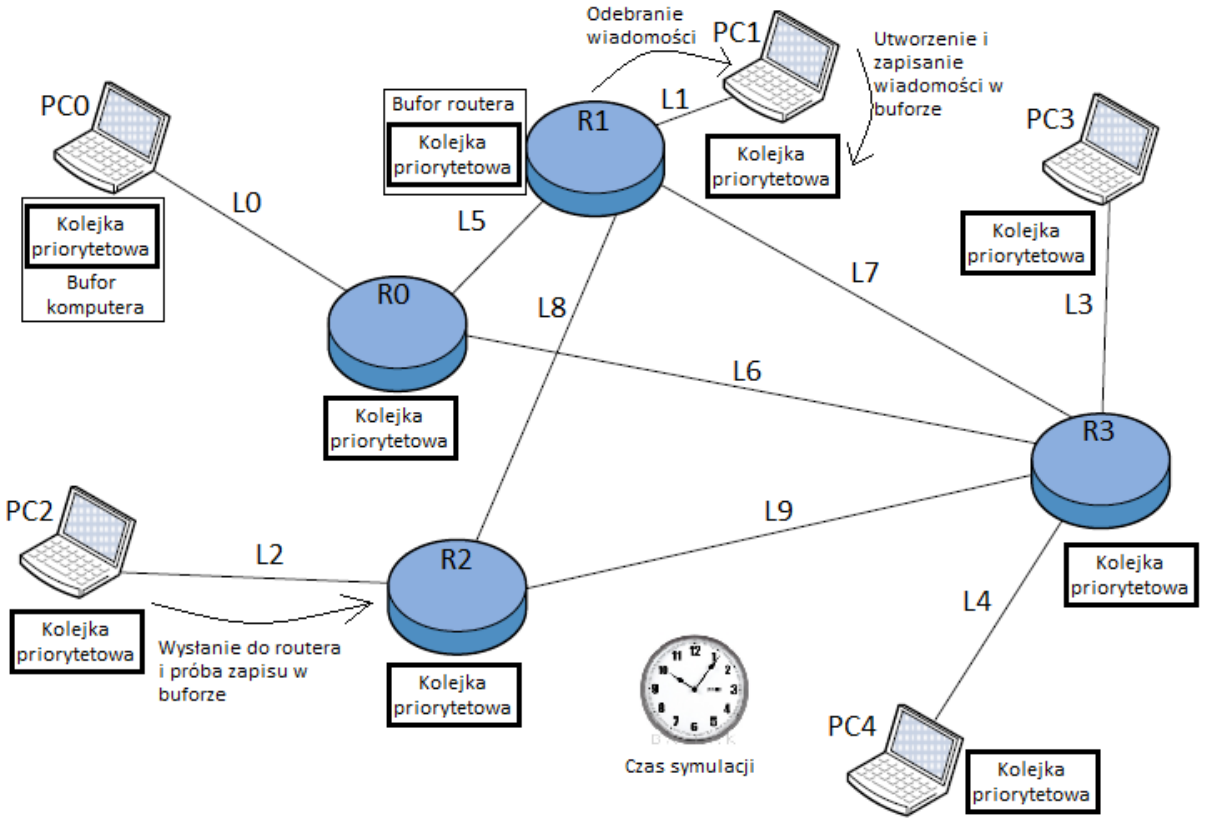
Wojciech Tyczyński

Symulacja cyfrowa

Zadanie 2

26.04.2014

Rys. 1. Schemat sieci



Według powyższego rysunku wyznaczone zostały tablice routingu. Przebieg wiadomości przez sieć wygląda następująco:

$$PC_0 \leftrightarrow R_0 \leftrightarrow R_1 \leftrightarrow PC_1$$
$$PC_0 \leftrightarrow R_0 \leftrightarrow R_1 \leftrightarrow R_2 \leftrightarrow PC_2$$
$$PC_0 \leftrightarrow R_0 \leftrightarrow R_3 \leftrightarrow PC_3$$
$$PC_0 \leftrightarrow R_0 \leftrightarrow R_3 \leftrightarrow PC_4$$
$$PC_1 \leftrightarrow R_1 \leftrightarrow R_2 \leftrightarrow PC_2$$
$$PC_1 \leftrightarrow R_1 \leftrightarrow R_3 \leftrightarrow PC_3$$
$$PC_1 \leftrightarrow R_1 \leftrightarrow R_3 \leftrightarrow PC_4$$
$$PC_2 \leftrightarrow R_2 \leftrightarrow R_3 \leftrightarrow PC_3$$
$$PC_2 \leftrightarrow R_2 \leftrightarrow R_3 \leftrightarrow PC_4$$
$$PC_3 \leftrightarrow R_3 \leftrightarrow PC_4$$

Tab. 1. Opis metod realizujących zmianę stanu systemu

Metoda	Opis metody	Atrybuty wejściowe	Modyfikowane zmienne modelu
`bool add_to_queue (Message*)`	Metoda klasy `Buffer`. Dodawanie wiadomości do bufora. Metoda zwraca wartość logiczną informującą, czy operacja się powiodła.	- wskaźnik do zapisywanej wiadomości	- kolejka bufora - rozmiar bufora
`bool save_msg (Message*)`	Metoda wirtualna. W przypadku routerów wywołuje `add_to_queue`, dla komputerów automatycznie uznaje wiadomość za dostarczoną.	- wskaźnik do zapisywanej wiadomości	- brak (metoda pośrednicząca)
`bool save_msg_to_buffer(Message*)`	Zapisuje wiadomość w buforze komputera. Wywołuje do tego funkcję `add_to_queue`.	- wskaźnik do zapisywanej wiadomości	- brak (metoda pośrednicząca)
`bool send_msg()`	Funkcja przeznaczona do obsługi wysyłania wiadomości z komputera lub routera. Wywołuje metodę `add_msg_to_link`.	- brak	- brak (metoda pośrednicząca)
`bool send_msg_to_device(int)`	Wywołuje `save_msg` na najbliższym urządzeniu odbiorczym oraz usuwa wiadomość z łącza.	- wartość oznaczająca wyjście	- wskaźnik do wiadomości w łączu - czas zajętości łącza
`Message *get_msg()`	Metoda klasy `Buffer`. Pobieranie wiadomości z bufora.	- brak	- kolejka bufora - rozmiar bufora
`void add_msg_to_link (int)`	Dodawanie wiadomości do łącza.	- wartość oznaczająca wyjście łącza (0 lub 1)	- wskaźnik do wiadomości w łączu
`void deliver_message ()`	Modyfikuje istotne zmienne modelu po odebraniu wiadomości przez komputer docelowy.	- brak	- czas dostarczenia wiadomości - liczba dostarczonych wiadomości

Opis narzędzia do automatycznej generacji dokumentacji – Doxygen.

Doxygen jest wieloplatformowym generatorem dokumentacji dla takich języków jak C++, C, Java oraz inne. Przy jego pomocy możliwe jest wygenerowanie dokumentacji w formacie HTML, PDF (przy użyciu LaTeX) a także innych. Oprócz samej struktury plików oraz klas i funkcji w formie tekstowej możliwe jest również zobrazowanie niektórych zależności graficznie, np. dzięki **Graphviz** – zestawu narzędzi do tworzenia diagramów za pomocą grafów. Konfiguracja przeglądania kodu oraz generacji danych wyjściowych zawarta jest w odpowiednio sformułowanym pliku tekstowym. Możliwa jest jego ręczna edycja lub generacja/modyfikacja przez narzędzie graficzne o nazwie **Doxywizard**. Utworzenie dokumentacji jest możliwe albo poprzez wywołanie komendy `doxygen <doxy.cfg>`, albo również z poziomu narzędzia Doxywizard.
Sama dokumentacja generowana jest na podstawie odpowiednio sformułowanych komentarzy (oczywiście zgodnych formatem z komentarzami C++) w newralgicznych miejscach kodu, takich jak deklaracje zmiennych czy funkcji. Możliwe jest tworzenie komentarzy według kilku koncepcji oraz z wykorzystaniem różnych znaczników wskazujących, czego ma dotyczyć dany fragment opisu. Dzięki wykorzystaniu dodatku **DoxygenComments** w środowisku **Visual Studio**, składnia Doxygena zostaje dodatkowo pokolorowana, co doprowadza do większej czytelności dokumentacji zawartej w komentarzach. Poniżej przedstawione zostały przykłady z rozwiązań zawartych w kodzie autora.

```
/*****************************************//**
 * \file      sc.cpp
 * \brief     Computer Simulation.
 * \author    Wojciech Tyczyński (tyczynskiwojciech@gmail.com)
 * \date      2014-04-13
 *****************************************/
```

Powyższy fragment służy do opisu pliku. Jak widać, plik można opisać za pomocą kilku znaczników:

- `\file` – nazwa pliku,
- `\brief` – krótki opis pliku,
- `\author` – autor pliku,
- `\date` – data utworzenia lub ostatniej modyfikacji (w zależności od przyjętej konwencji).

```
/*! \brief Time
 *
 * Class for time implementation.
 *
 */
```

W powyższy sposób można wprowadzać opisy definiowanych klas. Opis taki znajduje się przed definicją klasy. Znacznik `\brief` oznacza krótki opis, następnie (już bez specjalnego znacznika) możliwe jest wprowadzenie bardziej rozwiniętego opisu klasy.

```
double actual_time; //!< Actual time of simulation.
```

W powyższy sposób można wprowadzić krótki opis dla danej zmiennej.

```
//! Constructor.
Time();
```

Taki komentarz powoduje wprowadzenie opisu do funkcji znajdującej się w najbliższej niepustej linii za komentarzem.