ZTiT

Zakład Teleinformatyki i Telekomutacji

Transport i sterowanie

Instrukcja

Projekt: Emulacja sieci transportowej

ZTiT. Zakład Teleinformatyki i Telekomutacji
Instytut Telekomunikacji
Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska, Październik 2010

Założenia

Przedmiotem projektu jest zaimplementowanie systemu emulującego sieć transportową, składającą się z płaszczyzny transportu i płaszczyzny sterowania, realizowaną w oparciu o wskazaną technologię transportową. Celem emulacji jest zilustrowanie architektury sieci transportowej oraz procedur zestawiania trwałych i komutowanych połączeń transportowych.

Implementację systemu należy przeprowadzić na platformie aplikacyjnej .NET Framework w środowisku systemu operacyjnego Windows. Jako języka programowania należy użyć języka C#, a jako środowiska rozwojowego narzędzia Microsoft Visual Studio w wersji 2010. Niezbędne oprogramowanie można pobrać ze strony Microsoft Academic Alliance www.msdnaa.pl.

Technologia transportowa

Zbiór technologii sieci transportowej rozważanych w projekcie obejmuje: IP (Internet Protocol), ATM (Asynchronous Transport Mode), SDH (Synchronous Digital Hierarchy), WDM/WSON (Wavelength Division Multiplexing / Wavelength Switched Optical Network), OTN (Optical Transport Network), MPLS/MPLS-TP (Multi-Protocol Label Switching / MPLS-Transport Profile), PBB/PBB-TE (Provider Backbone Bridge / PBB-Traffic Engineering).

Przygotowując się do realizacji systemu emulacji sieci transportowej, należy przeanalizować dokumenty IETF (Internet Engineering Task Force; www.ietf.org), ITU-T (International Telecommunications Union; www.itu.int), OIF (Optical Interworking Forum; www.oiforum.com), MEF (Metro Ethernet Forum; www.metroethernetforum.org) odnoszące się do wskazanej technologii transportowej.

W ramach projektu należy przygotować prezentację dotyczącą najważniejszych aspektów rozważanej albo innej wskazanej, mniej znanej technologii: przesłanek wprowadzenia, zakresu i stanu standaryzacji, struktury sygnałów, architektury sieci, scenariuszu wdrażania, itp. Prezentacja powinna się składać z jak najmniejszej liczby slajdów (najwyżej kilku).

Emulator sieci

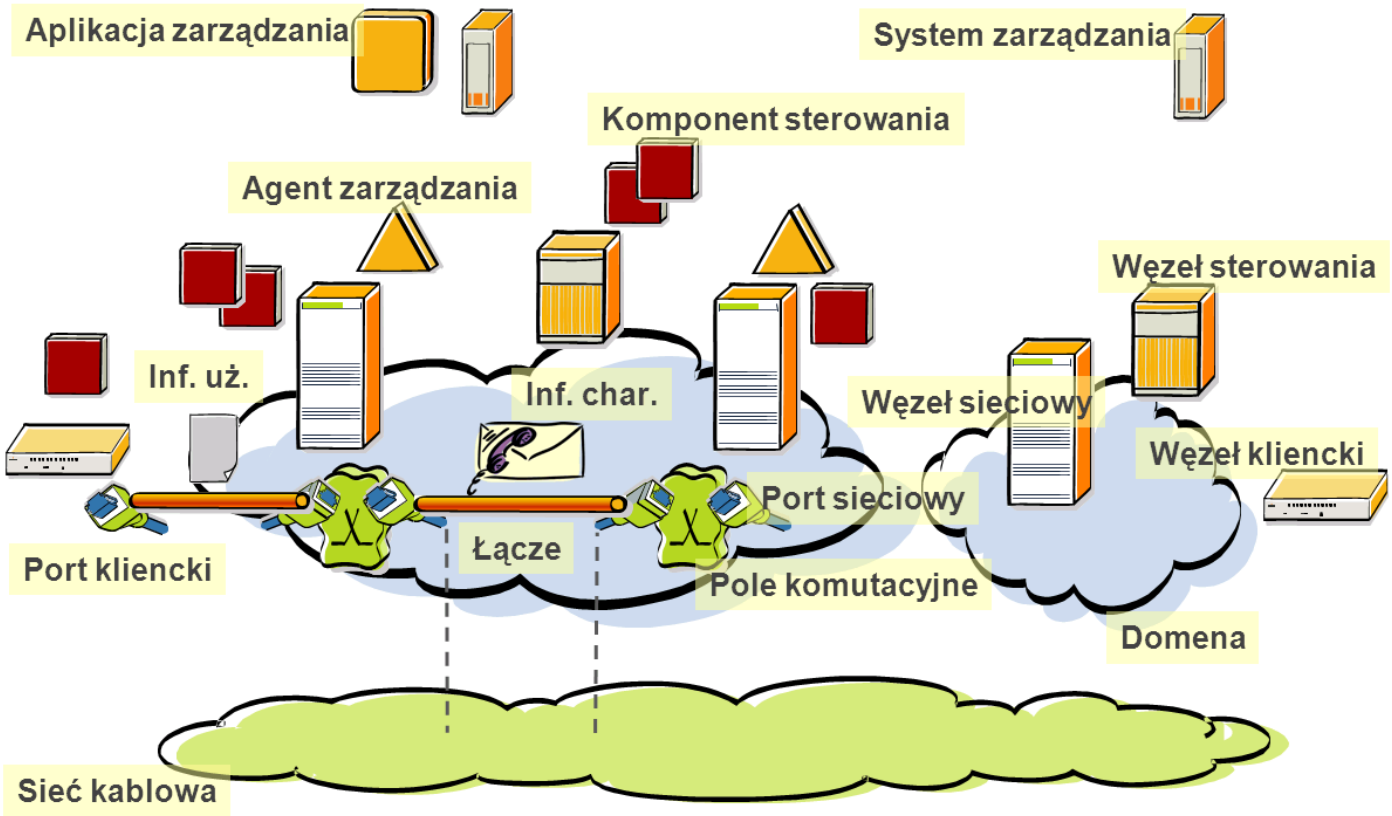
1. Płaszczyzna transportu

Należy zaimplementować obecność i funkcje płaszczyzny transportu sieci dla wskazanej technologii transportowej, pozwalające na obsługę informacji użytkowej i informacji charakterystycznej. Płaszczyzna transportu powinna się składać z dwóch typów węzłów transportowych – węzłów klienckich, będących źródłem informacji użytkowej oraz węzłów sieciowych, odpowiedzialnych za przesyłanie informacji użytkowej przy użyciu informacji charakterystycznej pomiędzy węzłami klienckimi.

Po pierwsze należy zdefiniować zakres i postać informacji użytkowej generowanej, wysyłanej i odbieranej przez węzły klienckie oraz informacji charakterystycznej generowanej, wysyłanej i

odbieranej przez węzły sieciowe, pamiętając o tym, że w ogólnym przypadku informacja charakterystyczna przenosi informację użytkową należącą do wielu strumieni.

Węzeł kliencki powinien być wyposażony w jeden lub więcej wyjściowych i wejściowych portów klienckich. Port wyjściowy ma móc generować i wysyłać, a porty wejściowy odbierać i interpretować informację użytkową. Należy udostępnić możliwość rozpoczynania i kończenia okresowego wysyłania informacji użytkowej przez zadany port. W tym celu węzeł kliencki powinien być wyposażony w konsolę.



Węzeł sieciowy powinien być wyposażony w pewną liczbę wejściowych i wyjściowych portów klienckich, w pewną liczbę wejściowych i wyjściowych portów sieciowych oraz w pole komutacyjne. Wejściowy port kliencki ma móc odbierać informację użytkową i generować informację charakterystyczną, a wyjściowy port kliencki ma móc interpretować informację charakterystyczną i wysyłać informację użytkową. Wyjściowy port sieciowy ma móc wysyłać, wejściowy port sieciowy odbierać informację charakterystyczną. Pole komutacyjne ma móc przekazywać informację charakterystyczną pomiędzy wejściowymi a wyjściowymi portami sieciowymi i/lub portami klienckimi. Konfiguracja każdego węzła sieciowego i klienckiego powinna być zadana w pliku konfiguracyjnym węzła.

Należy emulować łącza transportowe pomiędzy węzłami klienckimi a węzłami sieciowymi oraz pomiędzy parami węzłów sieciowych poprzez umożliwienie przesyłania informacji użytkowej lub charakterystycznej pomiędzy odpowiednią parą portów. Sposób połączenia portów powinien być zadany w pliku konfiguracyjnym węzła lub sieci.

2. Płaszczyzna zarządzania

Należy zaemulować obecność i funkcje płaszczyzny zarządzania, pozwalające na obsługę połączeń trwałych. Płaszczyzna zarządzania powinna się składać z agentów zarządzania w węzłach sieciowych oraz pełniącej role zarządcy aplikacji zarządzania, i stanowiącej główny element systemu zarządzania. Można się postarać, by zależnie w wypadku podziału płaszczyzny transportu na domeny, każdej z domen odpowiadał oddzielny system zarządzania. Podstawową operacją udostępnianą przez agenta zarządzania i dostępną z konsoli aplikacji zarządzania powinno być dodanie i usunięcie jednego lub więcej połączeń w polu komutacyjnym oraz odczyt zbioru połączeń w polu komutacyjnym wybranego węzła sieciowego. Należy udostępnić odpowiednie funkcje w konsoli aplikacji zarządzania. Model informacyjny na styku zarządca-agent powinien być zgodny z architekturą sieci transportowej przedstawioną w zaleceniu ITU G.805.

3. Płaszczyzna sterowania

Należy zaemulować obecność i funkcje płaszczyzny sterowania o architekturze zgodnej z zaleceniem ITU G.8080, pozwalające na obsługę połączeń półtrwałych i komutowanych. W skład płaszczyzny sterowania powinny wchodzić następujące typy komponentów sterowania: kliencki sterownik zgłoszeń, sieciowy sterownik zgłoszeń, katalog, sterownik połączeń, sterownik routingu, zarządca zasobów łącza, sterownik pakietów. Komponenty sterowania powinny być elementami urządzeń pełniących rolę klienckich oraz sieciowych węzłów transportowych albo samodzielnych sieciowych węzłów sterowania. Konfiguracja każdego węzła sterowania albo każdego węzła transportowego odnosząca się do komponentów sterowania powinna być zadana w pliku konfiguracyjnym węzła.

Należy udostępnić możliwość zestawiania i rozłączania przy użyciu konsoli węzła klienckiego komutowanych połączeń transportowych, zestawiania i rozłączania przy użyciu konsoli aplikacji zarządzania półtrwałych połączeń transportowych, oraz odtwarzania zestawionych połączeń transportowych w wypadku awarii sieciowego węzła transportowego. Należy spróbować umożliwić zestawianie połączeń w sieci składającej się z wielu domen. Należy dążyć do tego, by zależnie od struktury płaszczyzny sterowania, był emulowany hierarchiczny, federacyjny lub mieszany tryb zestawiania połączeń.

Realizacja projektu

Emulator powinien obejmować najważniejsze elementy architektury fizycznej sieci oraz najważniejsze komponenty architektury funkcjonalnej. Należy się starać odzwierciedlić niezależność elementów fizycznych oraz oddziaływanie komponentów funkcjonalnych. W tym celu każdy węzeł transportowy i każdy samodzielny węzeł sterowania, podobnie jak każdy inny element architektury fizycznej sieci odpowiadający pojedynczemu urządzeniu, należy spróbować zaimplementować jako oddzielny proces, który może być startowany i kończony niezależnie od pozostałych. Można również

rozważyć to, czy nie wprowadzić dodatkowego, pomocniczego, osobnego obiektu odpowiadającego za emulację całej sieci kablowej, również implementowanego jako oddzielny proces; jego wprowadzenie może uprościć opis konfiguracji sieci kablowej i zbioru łączy transportowych oraz emulowanie awarii systemów transmisyjnych. Do startowania zbiorów procesów należy wykorzystać mechanizm plików startowych systemu operacyjnego.

Konfiguracja fizycznych i funkcjonalnych bloków urządzenia powinna być zadana w pliku konfiguracyjnym odpowiadającym danemu urządzeniu, który powinien być interpretowany przy starcie procesu. Należy dążyć do tego, by maksymalnie ograniczyć ilość statycznie zapisanej informacji konfiguracyjnej, na rzecz dynamicznego pozyskiwania niezbędnej informacji od innych komponentów.

Przesyłanie informacji użytkowej i informacji charakterystycznej pomiędzy portami węzłów transportowych oraz przesyłanie informacji sygnalizacyjnej pomiędzy komponentami węzłów sterowania lub komponentami węzłów zarządzania można zrealizować przy użyciu mechanizmu serializacji i deserializacji obiektów oraz mechanizmu gniazd (*socket*) i wybranego protokołu warstwy transportowej; w wypadku warstwy sterowania do implementacji sterownika pakietów, a w wypadku warstwy transportowej do implementacji portów i sieci kablowej.

Realizacja emulatora powinna przebiegać w dwóch etapach (trwających około pięciu tygodni każdy). W pierwszym etapie należy zaimplementować emulator płaszczyzny transportowej i płaszczyzny zarządzania. Każda osoba powinna być odpowiedzialna za implementację podobnej liczby komponentów funkcjonalnych sieci zdefiniowanych dla tych płaszczyzn. Należy przygotować pokaz możliwości emulatora, ilustrujący zestawianie i rozłączanie pewnej liczby jednoczesnych połączeń transportowych poprzez konfigurację pól komutacyjnych węzłów sieciowych za pośrednictwem konsoli aplikacji zarządzania, oraz przesyłanie informacji użytkowej pomiędzy węzłami klienckimi.

W drugim etapie należy zaimplementować emulator płaszczyzny sterowania. Każda osoba powinna być odpowiedzialna za implementację podobnej liczby komponentów funkcjonalnych sieci zdefiniowanych dla tej płaszczyzny. Należy przygotować pokaz możliwości emulatora, ilustrujący zestawianie, rozłączanie, a potencjalnie również odtwarzanie po wystąpieniu awarii, pewnej liczby jednoczesnych połączeń transportowych poprzez wysłanie żądania za pośrednictwem konsoli węzła klienckiego, oraz przesyłanie informacji użytkowej pomiędzy węzłami klienckimi.

Pokazy należy przygotować dla zaproponowanej przykładowej struktury sieci transportowej odnoszącej się do poszczególnych płaszczyzn.

Sprawozdanie

W sprawozdaniu należy krótko opisać przyjęte założenia, architekturę oraz sposób użycia zaprojektowanego systemu. Sprawozdanie powinno mieć formę dokumentu w formacie pdf. Do sprawozdania należy dołączyć rozwiązanie i projekty Visual Studio zawierające kody źródłowe i kody wykonywalne programów oraz pliki konfiguracyjne i pliki startowe systemu.