Programowanie sieciowe

Protokoły sieciowe, model warstwowy DOD, protokoły TCP i UDP, adresy IP, porty, gniazda, operacje na adresach - klasa InetAddress i U RL, komunikacja poprzez sieć.

FUW

Protokoły sieciowe

- **Aby połączone w sieć komputery mogły komunikować się ze sobą, muszą posługiwać się wspólnym językiem. Takim "językiem" jest protokół sieciowy - specyfikacja określająca postać danych przesyłanych w sieci.**
- **Istnieją trzy podstawowe protokoły sieciowe:**
  - **NetBEUI**- opracowany w 1985 roku przez firmę IBM. Używany jest zwykle w małych, odizolowanych sieciach LAN typu peer to peer.
  - **IPX/SPX** - opracowany na początku lat siedemdziesiątych przez firmę Novell. Jest protokółem routowalnym i dlatego może być wykorzystany do budowy złożonych sieci. W protokole tym adres stacji jest generowany automatycznie poprzez połączenie adresu sieci oraz adresu sprzętowego karty sieciowej.
  - **TCP/IP** - najczęściej stosowany zestaw protokołów sieciowych, z których najważniejsze to TCP (Transmission Control Protocol) oraz IP (Internet Protocol). Protokół ten może być wykorzystany do budowy intranetu oraz sieci heterogenicznej - łączącej komputery pracujące na różnych platformach sprzętowo-systemowych.

Protokół TCP/IP

- **Protokół TCP/IP jest internetowym standardem, określającym podstawowe zasady wymiany informacji między komputerami.**
- **Protokół ten najczęściej przedstawiany jest jako system składający się z czterech warstw, którym odpowiadają określone elementy sprzętowe i programowe, biorące udział w sesji komunikacyjnej.**
- **Poszczególne warstwy tego protokołu reprezentuje model DOD (Department of Defense).**

Warstwa dostępu do sieci	ARP, protokoły łącza
Warstwa Internetu	IP, ICMP, IGMP
Warstwa hosta z hostem	TCP, UDP
Warstwa procesu/aplikacji	HTTP, FTP, Telnet, e-mail, itp.

Model warstwowy DOD

- **Warstwa dostępu do sieci jest najniższą warstwą modelu i to dzięki niej nasze dane zamienione w impulsy elektryczne lub świetlne biegną w sieci. W warstwie tej działa protokół rozróżniania adresów ARP (Address Resolution Protocol), który pozwala określić adres sprzętowy karty sieciowej odpowiadający danemu adresowi internetowemu.**
  - Polecenie arp -a umożliwia wyświetlenie tabeli zawierającej ostatnio przetłumaczone adresy IP oraz odpowiadające im adresy fizyczne.
- **Warstwa Internetu odpowiada za wybranie marszruty dla danych. Kluczowym protokołem tej warstwy jest IP , który odpowiada za adresowanie, dostarczanie i kierowanie pakietów danych. Korzysta on z usług innych protokołów, z których najważniejszy to ICMP (Internet Control Message Protocol), kontrolujący komunikację w sieci Internet. Dzięki niemu można uzyskać informacje o awariach sieci czy osiągalności hosta docelowego. Znanym zastosowaniem tego protokółu jest program ping.**

Model warstwowy DOD

- **Warstwa hosta z hostem - odpowiada za integralność przesyłanych danych. Zawiera dwa protokoły transportowe:**
  1. TCP (Transmission Control Protocol),
  2. UDP (User Datagram Protocol).

  Pierwszy z nich odpowiada za procesy przesyłania danych w sieci, wyposażony jest zatem w mechanizmy gwarantujące poprawność transmisji. Drugi natomiast zapewnia bezpołączeniowe usługi transmisji datagramów i nie zajmuje się sprawdzaniem, czy wszystkie pakiety danych dotarły do miejsca ich przeznaczenia.
- **Warstwa procesu/aplikacji - najwyższa warstwa modelu DOD, która pełni rolę interfejsu pomiędzy aplikacjami widocznymi dla użytkownika (przeglądarka, program pocztowy) a usługami sieci. Znajdują się w niej protokoły związane z najbardziej powszechnymi usługami internetowymi, takimi jak:**
  - HTTP, FTP, Telnet, NFS, DNS, SMTP.

Protokół TCP

- **Protokół transportowy TCP odpowiada za to, aby dane dotarły do odbiorcy w takim stanie i kolejności, w jakich były wysłane.**
  - jest protokółem zorientowanym na połączenie;
  - posiada wbudowany mechanizm kontroli błędów.
- **Działanie TCP polega na utworzeniu wirtualnego obwodu pomiędzy nadawcą i odbiorcą. Zanim nastąpi przesyłanie danych, musi zostać nawiązana sesja TCP/IP. Sesja taka nosi nazwę potrójnego uścisku dłoni. Można w niej wyróżnić trzy etapy:**
  - klient wysyła żądanie połączenia ze zdalnym komputerem (serwerem),
  - serwer odpowiada potwierdzeniem odbioru,
  - klient również potwierdza odbiór i wirtualny obwód zostaje ustalony.
- **Po ustaleniu obwodu dane mogą się przemieszczać w obu kierunkach równocześnie. Taki rodzaj transmisji nazywany jest transmisją pełnodupleksową (full-duplex). Dzięki niej komunikaty o błędach transferu mogą być natychmiast wysłane do zdalnego komputera.**

Protokół UDP

- **Wszystkie implementacje TCP/IP muszą również obsługiwać prostszy protokół transportowy User Datagram Protocol.**
- **UDP jest protokółem bezpołączeniowym - zarządza transmisją bez połączenia zwrotnego.**
- **Protokół ten pozbawiony jest kontroli transmisji danych i nie gwarantuje, że wysłane pakiety danych dotrą do miejsca przeznaczenia. Dzięki temu transmisja według jego reguł jest znacznie szybsza.**
- **Protokół UDP jest używany w sytuacjach, w których protokół TCP nie jest konieczny, tzn. wtedy, gdy uszkodzenie lub zagubienie części danych nie wyrządzi znacznych szkód. Najczęściej stosuje się go w aplikacjach działających w czasie rzeczywistym, np. podczas przesyłania na żywo transmisji dźwięku lub obrazu. Korzystanie w takim przypadku z protokółu TCP spowodowałoby nadmiar niepotrzebnych informacji.**

Adresy IP

- **Podstawą komunikacji komputerów w Internecie lub intranecie jest niepowtarzalny adres IP każdego komputera.**
- **Obecnie używana wersja protokołu IP stosuje adresy, które są 32-bitowymi liczbami binarnymi.**
- **Dla ułatwienia adres dzieli się na cztery części zwane oktetami, które są oddzielone kropkami. Każdy oktet zawiera osiem bitów, które dla uproszczenia przekształca się w liczbę dziesiętną z zakresu 0-255.**
  - Na przykład adres IP: 11000011 10100100 11010101 01000111 można zapisać w postaci: 195.164.213.71. Taki zapis nazywany jest kropkowaną notacją dziesiętną.
- **Ważnym, zastrzeżonym adresem jest np. 127.0.0.1, który jest adresem lokalnego komputera - stąd jego nazwa localhost.**
- **Gdy dane w postaci pakietów są przesyłane siecią, każdy nagłówek pakietu zawiera adres nadawcy i odbiorcy.**

Porty

- **Port jest adresem wewnętrznym, który zapewnia interfejs pomiędzy aplikacją a protokółem warstwy transportowej.**
- **Każde połączenie TCP odbywa się przez port.**
- **Port jest reprezentowany przez 16-bitową liczbę od 1 do 65 535. Liczba ta jest wykorzystywana do tego, aby program sieciowy wiedział, z którą usługą działającą na serwerze ma się kontaktować. Na przykład gdy użytkownik odbiera pocztę elektroniczną, jego program pocztowy upomina się o nią na porcie serwera SMTP o numerze 25.**
- **Porty od 1 do 1023 to tzw. dobrze znane porty (well-know ports), które są przypisane określonym usługom. Na przykład:**
  - port 21 odpowiada protokółowi FTP,
  - port 23 - Telnet,
  - port 80 protokół HTTP.
- **Numery portów przyporządkowywane są przez Internet Assign Numbers Authority (IANA).**

Podstawy gniazd

- **Gniazdo jest mechanizmem komunikacyjnym, umożliwiającym transmisję danych pomiędzy urządzeniami w sieci.**
- **Wyróżniamy gniazda strumieniowe, komunikujące się za pomocą protokołu TCP oraz gniazda datagramowe, działające w oparciu o protokół UDP.**
- **Wszystkie rodzaje gniazd są reprezentowane przez klasy znajdujące się w pakiecie java.net.**
- **Dla gniazd TCP mamy dwie klasy:**
  - Socket - do tworzenia gniazd klienckich,
  - ServerSocket - do tworzenia gniazd serwerowych.
- **Gniazda UDP tworzymy za pomocą klasy DatagramSocket, która jest wykorzystywana zarówno przez nadawcę jak i odbiorcę danych. Do przechowywania danych tworzone są obiekty klasy DatagramPacket.**

Podstawy gniazd

- **Pakiet java.net zawiera klasy umożliwiające przeprowadzanie typowych operacji sieciowych. Do najważniejszych zaliczamy:**
- Socket, ServerSocket,
- DatagramSocket,
- DatagramPacket
- **Gniazda klienckie pozwalają na wykonywanie operacji otwierania połączeń z odległymi komputerami, wysyłania i odbierania danych oraz zamykania połączeń.**
- **Gniazda serwerowe umożliwiają zaimplementowanie prostych serwerów w Javie.**
- **Po utworzeniu gniazda serwera, rozpoczyna ono nasłuchiwanie na określonym porcie, czekając na połączenie pomiędzy zdalnym klientem i lokalnym serwerem. Po nawiązaniu połączenia tworzone jest gniazdo klienckie, służące do wysyłania i odbierania danych. Po zakończeniu transakcji gniazdo klienckie zostaje zamknięte, a gniazdo serwera wraca do nasłuchu.**

Operacje na adresach

- **Klasa InetAddress opisuje adres komputera w sieci poprzez:**
  1. nazwę/domenę, np. www.fuw.edu.pl oraz
  2. numer IP, np. 193.0.81.28.
- **Istnieje szereg metod statycznych klasy InetAddress, które są przeznaczone do tworzenia obiektów klasy, brak jest bowiem konstruktorów. Podstawowe metody to:**
  - InetAddress.getByName(String nazwa);
    // tworzy obiekt bazując na podanej nazwie komputera lub adresie
  - InetAddress.getAllByName(String nazwa);
    // wykorzystywana wtedy, gdy komputer o danej nazwie ma wiele adresów IP. Zwracana jest wtedy tablica obiektów typu InetAddress.
  - InetAddress.getLocalHost();
    // tworzy obiekt reprezentujący adres komputera lokalnego
- **Metody te muszą zawierać deklaracje lub obsługę wyjątku UnknownHostException, który powstaje w przypadku braku identyfikacji komputera o podanej nazwie lub adresie.**

FUW

Przykład 1

Program wyświetla informacje o adresach sieciowych wybranych komputerów.

```
import java.net.*;

public class Adresy {
public static void main(String args[]) {
 try {
   InetAddress a0 = InetAddress.getLocalHost();
   String nazwa = a0.getHostName();
   System.out.println("Adres komputera "+nazwa+": " +a0);
   InetAddress a1 =
InetAddress.getByName("tempac.fuw.edu.pl");
   System.out.println("Adres komputera tempac to: "+a1);
   InetAddress a2[] = InetAddress.getAllByName("waw.edu.pl");
   System.out.println("Adres komputera waw.edu.pl to:");
   for(int i=0; i<a2.length; i++) {
   System.out.println(a2[i]);
   }
 } catch (UnknownHostException e) {e.printStackTrace();}
}
}
```

FUW

Klasa URL

- **Inną klasą wykorzystywaną w Javie do adresowania komputerów jest klasa URL oraz jej pochodne:**
  - URLClassLoader,
  - URLConnection,
  - URLDecoder,
  - URLEncoder,
  - URLStreamHandler.
- **URL czyli Uniform Resource Locator jest specjalną formą adresu zasobów w sieci. URL posiada dwa podstawowe elementy: identyfikator protokołu oraz nazwę zasobów. Identyfikator protokołu to np. http, ftp, gopher, rmi czy jdbc. Nazwę zasobów stanowią takie elementy jak: nazwa hosta, nazwa pliku, numer portu, nazwa odwołania (w danym pliku).**
- **Tworząc obiekt klasy URL otrzymujemy gotowy wskaźnik, który jest wykorzystywany przez liczne metody Javy (np. otwieranie obrazka getImage(), tworzenie połączenia w JDBC - Connection).**

Konstruktory klasy URL

- **W odróżnieniu od klasy InetAddress tworzenie obiektów klasy URL odbywa się poprzez wykorzystanie jednego z jej licznych konstruktorów.**
- **Każdy z nich związany jest z koniecznością obsługi wyjątku MalformedURLException powstającym w przypadku problemów z identyfikacją wskazanego w wywołaniu konstruktora protokołu.**
- **Przykładowe konstruktory:**
  - URL(String adres),
  - URL(String protokół, String host, int port, String plik)
- **Klasa URL zawiera szereg metod umożliwiających filtrację adresu, a więc pobranie nazwy protokołu - getProtocol(), nazwy komputera - getHost(), pliku - getFile() czy numeru portu - getPort().**
- **Dodatkowo klasa URL zawiera metody umożliwiające wykonywanie połączenia z hostem - tworzenie gniazda i przesłanie danych.**

Przykład 2

Program umożliwia pobranie kodu źródłowego wskazanej strony WWW i wyświetlenie go na ekranie.

```
import java.net.*;
import java.io.*;

public class Pobierz {
public static void main(String args[]) {
URL url;
String tekst;
try {
url = new URL("http://www.fuw.edu.pl/");
InputStreamReader in = new
                       InputStreamReader(url.openStream());
BufferedReader br = new BufferedReader(in);
while( (tekst=br.readLine()) !=null) {
System.out.println(tekst);
}
} catch (Exception e) {e.printStackTrace();}
}
}
```

FUW

Komunikacja poprzez sieć

- **Do komunikacji poprzez Internet programy Javy wykorzystują protokoły TCP i UDP.**
- **Klasy URL oraz Socket i ServerSocket wykorzystują Transfer Control Protocol.**
- **Klasy takie jak DatagramPacket, DatagramSocket, oraz MulticastSocket korzystają z User Datagram Protocol.**
- **W pakiecie java.net zdefiniowane są jeszcze inne klasy, z których warto przytoczyć te związane z autoryzacją połączeń i nadawaniem uprawnień: Authenticator, NetPermission, PasswordAuthentication, SocketPermission.**
- **W aplikacjach klient-serwer, serwer dostarcza określonej usługi np. przetwarza zapytanie skierowane do bazy danych, zapisuje serię obrazów diagnostycznych.**
- **Klient wykorzystuje usługi świadczone przez serwer i jest odpowiedzialny za żądanie usługi oraz obsługę wyników.**

Komunikacja poprzez sieć

- **Funkcje pełnione przez każdą ze stron można ująć następująco:**
  - połączenie z urządzeniem zdalnym,
  - wysyłanie danych,
  - odbiór danych,
  - zamknięcie połączenia,
  - przywiązanie (binding) portu (dla danej aplikacji na danym hoście)
  - nasłuch,
  - akceptacja połączeń na danych portach.
- **Pierwsze cztery funkcje są właściwe dla klienta, serwer rozszerza je o dodatkowe trzy.**
- **W Javie obsługę klienta, a więc realizację pierwszych czterech funkcji dostarcza klasa Socket; dla serwera przeznaczono klasę ServerSocket.**
- **Aby otworzyć gniazdo, trzeba zadeklarować egzemplarz klasy Socket, a następnie utworzyć go za pomocą konstruktora Socket.**

Klasa Socket

- **Klasa Socket posiada szereg różnych konstruktorów, z których najbardziej popularne są dwa:**
  - `Socket(String host, int port)`
    ```
    // gdzie host to tekst oznaczający nazwę hosta
    // port - numer portu (0-65535)
    ```
  - `Socket(InetAddress address, int port)`
    ```
    //gdzie address to obiekt klasy InetAddress będący
    //adresem IP lub nazwą hosta
    ```
- **Klasa Socket posiada wiele metod umożliwiających m.in. uzyskanie informacji związanych z obiektem tej klasy, takich jak:**
  - getLocalAddress() - zwraca lokalny adres, do którego dowiązane jest gniazdo,
  - getLocalPort() - zwraca lokalny port, do którego dowiązane jest gniazdo,
  - getInetAddress() - zwraca zdalny adres, do którego gniazdo jest podłączone,
  - getPort() - zwraca zdalny numer portu, do którego gniazdo jest podłączone.

Klasa Socket

- **Ponieważ połączenie wykonywane przez obiekt klasy Socket może nie powieść się z różnych przyczyn (np. nie znany host) konieczna jest więc obsługa wyjątków. Klasyczny fragment kodu obsługi klasy Socket pokazano poniżej:**

```
try {
 Socket gniazdo = new Socket("www.fuw.edu.pl", 80);
}
catch (UnknownHostException e) {
  System.err.println(e);
}
catch (IOException e) {
   System.err.println(e);
}
```

- **W powyższym przykładzie podjęta jest próba połączenia z serwerem www.fuw.edu.pl na porcie 80. Jeśli nazwa hosta jest nieznana lub serwer nazw nie działa, to zostanie zwrócony wyjątek nieznanego hosta - UnknownHostException. Jeśli z innych przyczyn nie uda się uzyskać połączenia, zostanie zwrócony wyjątek IOException.**

FUW

Przykład 3

Przykładem programu implementującego klasę Socket może być prosty skaner portów, na których działają aplikacje (serwery).

```
import java.net.*;
import java.io.*;

public class SkanerPortow {
public static void main(String[] args) {
Socket gniazdo;
String host = "localhost";
if (args.length > 0) {
host = args[0]; //jeśli nie podano argumentu programu
}               // hostem bedzie komputer lokalny
for (int n = 0; n < 1024; n++) {
try {
gniazdo = new Socket(host, n);
System.out.println("Otwarty jest port "+n+" hosta: "+ host);
}
catch (UnknownHostException e)
{System.err.println(e);break;}
catch (IOException e) { }
}
}
```

Przykład 4

Program łączy się z serwerem czasu a następnie wyświetla ten czas na ekranie.

```
import java.net.*;
import java.io.*;

public class Zegar {
public static void main(String[] args) {
Socket gniazdo;
String host = "info.cyf-kr.edu.pl";
BufferedReader strumienCzasu;
if (args.length > 0) {
host = args[0];
}
try {
gniazdo = new Socket(host, 13);
strumienCzasu = new BufferedReader(new
                  InputStreamReader(gniazdo.getInputStream()));
String czas = strumienCzasu.readLine();
System.out.println("Na "+host+" jest: "+czas);
}
catch (UnknownHostException e) {System.err.println(e);}
catch (IOException e) {System.err.println(e);}
}
}
```

Podsumowanie kursu

- **Java jest stosunkowo młodym językiem programowania, który szybko zyskał popularność i uznanie programistów na całym świecie.**
- **Zadaniem pierwszej części kursu Javy było zapoznanie jego uczestników z podstawami języka.**
- **Nie zostały omówione jednak zagadnienia, które z pewnością są warte uwagi. Do takich tematów można zaliczyć:**
  - pisanie programów sieciowych z interfejsem graficznym;
  - programowanie z wykorzystaniem pakietu Swing;
  - zastosowania technologii JSP;
  - podstawy JavaBeans;
  - metody rodzime - JNI;
  - Java i bazy danych - podstawy technologii Java Database Connectivity
  - wprowadzenie do technologii Java 2 Micro Edtion
  - zagadnienia bezpieczeństwa aplikacji Javy.