CMPU4021 Distributed Systems Lab Notes - Week 1

Intro to Networking in Java and Python

Java Networking APIs

- Through the classes in `java.net`, Java programs can use TCP or UDP to communicate over the Internet

- The `URL, URLConnection, Socket, and ServerSocket` classes all use TCP to communicate over the network

- The `DatagramPacket, DatagramSocket, and MulticastSocket` classes are for use with UDP.

java.net

- Core package *java.net* provides the classes for implementing networking application

- Using these classes, the network programmer can communicate with any server on the Internet or implement his or her own Internet server.

Programmatic Access to Network Parameters

- Systems often run with multiple active network connections, such as wired Ethernet, 802.11 b/g (wireless), and bluetooth.
- Some applications might need to access this information to perform the particular network activity on a specific connection.
- The `java.net.NetworkInterface` class provides access to this information.

Network Interface

- A network interface is the point of interconnection between a computer and a private or public network.

- `NetworkInterface` is useful for a multi-homed system, which is a system with multiple NICs.

- Using `NetworkInterface`, you can specify which NIC to use for a particular network activity.

Class *java.net.InetAddress*

- This class represents an Internet Protocol (IP) address. An IP address is either a 32-bit or 128-bit unsigned number used by IP.

- Static method *getByName* of this class uses DNS (Domain Name System) to return the Internet address of a specified host *name* as an *InetAddress* object.

  - `public static InetAddress` **`getByName`**`(String host) throws UnknownHostException`
    - Determines the IP address of a host, given the host's name. The host name can either be a machine name, such as "java.sun.com", or a textual representation of its IP address. If a literal IP address is supplied, only the validity of the address format is checked.

Class *InetAddress*

```
String host;
try {
   InetAddress address = InetAddress.getByName(host);
   System.out.println("IP address: " + address.toString());
}
catch (UnknownHostException e){
   System.out.println("Could not find " + host);
}
```

Working with URLs in Java

- Java programs can use a class called `URL` in the java.net package to represent a URL address

- Creating an *absolute* URL
  - An absolute URL contains all of the information necessary to reach the resource in question.

```
URL myURL = new URL("http://example.com/");
```

Creating a URL Relative to Another

- A relative URL contains only enough information to reach the resource relative to (or in the context of) another URL

- You can create a URL object from a relative URL specification. For example, if you know two URLs at the site *example.com*:

```
http://example.com/pages/page1.html
http://example.com/pages/page2.html
```

- You can create URL objects for these pages relative to their common base URL: http://example.com/pages/ like this:

```
URL myURL = new URL("http://example.com/pages/");
URL page1URL = new URL(myURL, "page1.html");
URL page2URL = new URL(myURL, "page2.html");
```

Creating a URL Relative to Another

The general form of URL constructor is:

`URL(URL` *`baseURL`*`, String` *`relativeURL`*`)`

- The first argument is a URL object that specifies the base of the new URL.
- The second argument is a String that specifies the rest of the resource name relative to the base.
- If `baseURL` is null, then this constructor treats `relativeURL` like an absolute URL specification.
- If `relativeURL` is an absolute URL specification, then the constructor ignores `baseURL`.

Parsing a URL

- The `URL` class provides several methods that let you query URL objects.
- You can get the protocol, authority, host name, port number, path, query, filename, and reference from a URL using accessor methods (all are listed in Java docs)

`getProtocol`

Returns the protocol identifier component of the URL.

`getAuthority`

Returns the authority component of the URL.

`getHost`

Returns the host name component of the URL.

`getPort`

Returns the port number component of the URL. The getPort method returns an integer that is the port number. If the port is not set, getPort returns -1.

`getPath`

Returns the path component of this URL.

PYTHON NETWORK PROGRAMMING

Python Networking

- Python plays an essential role in network programming.

- The standard library of Python has full support for network protocols, encoding, and decoding of data and other networking concepts.

Python Networking

- There are two levels of network service access in Python:
  - Low-Level Access
  - High-Level Access

- Low-Level Access
  - Programmers can use and access the basic socket support for the operating system using Python's libraries
  - Allows to implement both connection-less and connection-oriented protocols for programming.

- High-Level Access
  - Application-level network protocols (HTTP, FTP, etc.) can also be accessed using high-level access provided by Python libraries.

References

- https://docs.oracle.com/javase/tutorial/networking
- https://www.w3schools.in/python/network-programming