PEMROGRAMAN SOCKET DENGAN JAVA DALAM MENGEMBANGKAN *SOFTWARE* DENGAN ARSITEKTUR *CLIENT SERVER*

Ahmad Nurul Fajar
Dosen FASILKOM - UIEU
nurul.fajar@lecturer.indonusa.ac.id

Abstrak

Pemrograman socket atau *Socket Programming* adalah salah satu cara yang dapat digunakan untuk mengembangkan *software* dengan arsitektur *Client Server*. Aplikasi yang dikembangkan menggunakan teknik seperti ini banyak digunakan untuk dunia internet, *security* dan para internet hacker khususnya. Pemahaman tentang protokol, socket, port, OSI layer, API, dan pengalamatan sangat diperlukan bila ingin melakukan teknik ini. Bahasa Java merupakan salah satu bahasa yang sangat mendukung untuk membuat pemrograman *socket*

Kata Kunci: Pemrograman Socket, *Java, Client Server*

Pendahualuan

Dalam mengembangkan perangkat lunak *(software)* dengan menggunakan arsitektur client server ada beberapa teknik yang bisa digunakan, pemilihan teknik tersebut disesuaikan dengan kebutuhan.

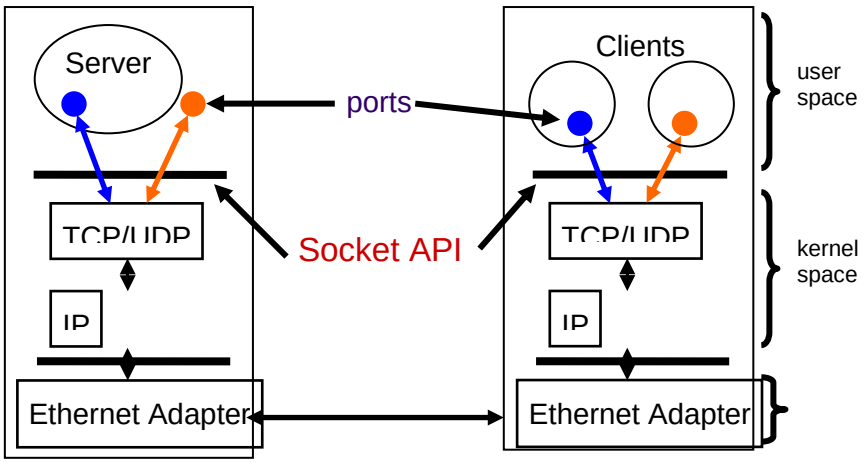
Arsitektur *Client server* dapat dikelompokkan menjadi :

- Arsitektur *two tiered*
  - Merupakan arsitektur *client server* dengan lingkungan tradisional yang biasanya hanya melibatkan *client* sebagai user *interface* dan *server* sebagai pengolah database yang sering disebut dengan database server
- Arsitektur *three tiered*
  - Merupakan perluasan dari arsitektur two tiered dengan menambahkan komponen yang disebut "*middleware*", karena *middleware* inilah yang memainkan peranan penting dalam arsitektur *three tiered*. Komponen yang berada di antara *client* dan *server* ini sering disebut dengan *business logic*. Dengan adanya komponen *business logic* dapat mengurangi beban database server karena terjadi sharing pekerjaan dengan *business logic*
- Arsitektur *many tiered*
  - Pengembangan lebih lanjut dari kedua arsitektur sebelumnya. Aplikasi didistribusikan ke lebih dari tiga platform yang biasanya dilakukan dengan membagi proses bisnis.

Pemrograman Socket

Socket adalah sebuah abstraksi perangkat lunak yang digunakan sebagai suatu terminal dari suatu hubungan antara dua mesin atau proses yang saling berinterkoneksi. Pada masing-masing mesin tersebut harus terpasang *socket.* Bila

kita berbicara tentang *client server* identik dengan pemrograman jaringan (*network*). *Socket* dibutuhkan sehingga antara *client* dan *server* bisa saling berhubungan.

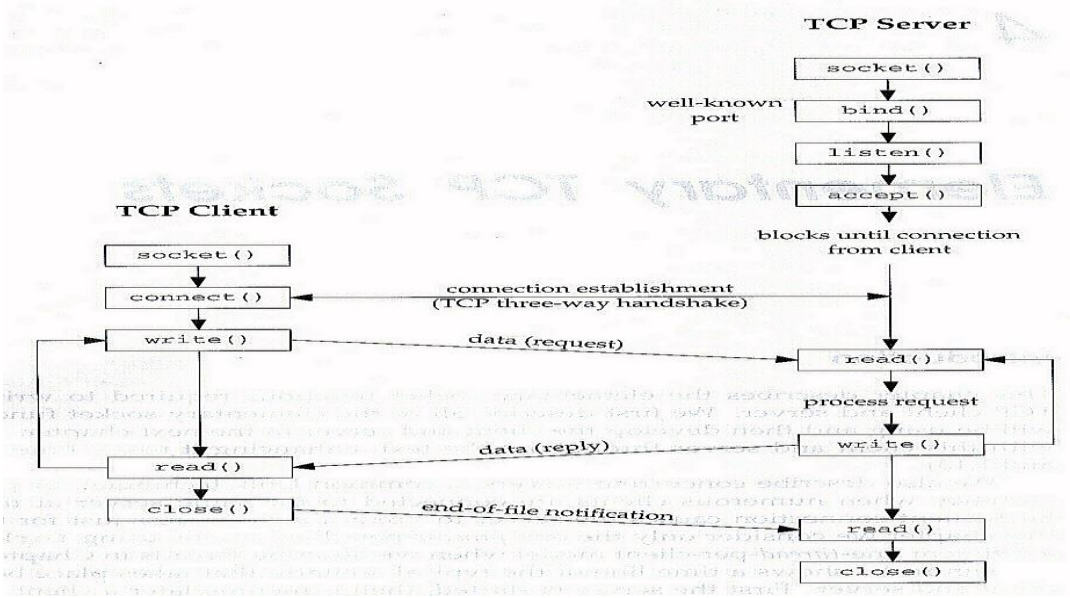


Sumber: www.javaworld.com

Gambar 1. hubungan *server-client*

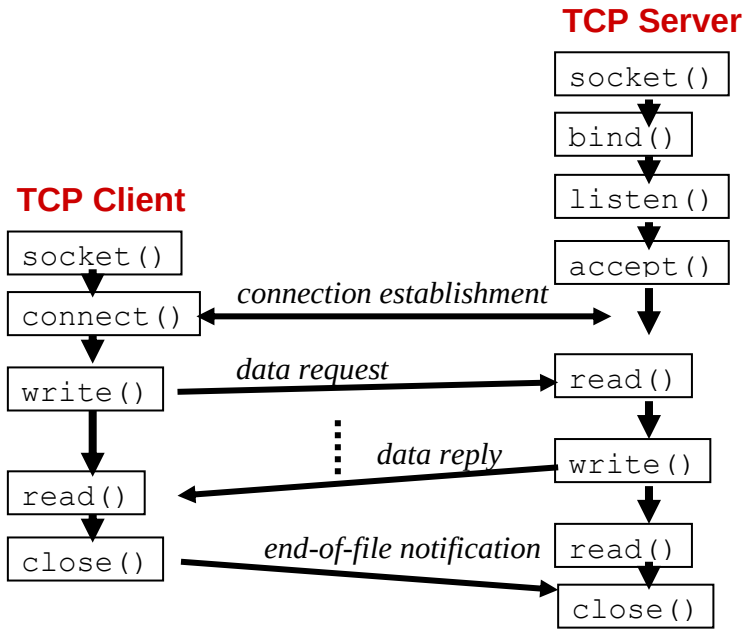
Pembahasan

Kita dapat menggunakan bahasa pemrograman Java, C++/C, Visual Basic.net dan bahasa pemrograman lainnya untuk membangun aplikasi dengan menggunakan pemrograman socket. Hal yang harus diperhatikan adalah API (*Application Programming Interface*) Karena API merupakan *interface* yang akan digunakan dalam jaringan



Sumber: www.javaworld.com

Gambar 2. Penciptaan socket



Sumber: www.javaworld.com

Gambar 3 Interaksi TCP Client-Server

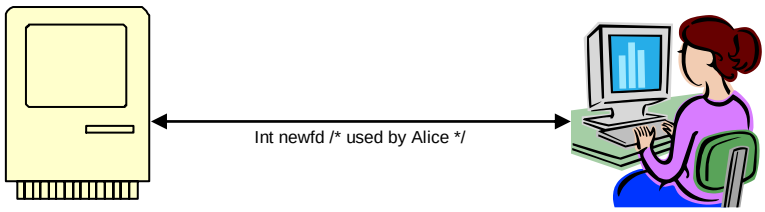
Address, Port dan *Socket*

Ilustrasi dari ketiga kata di atas seperti berikut :

a) Misalkan kita adalah "aplikasi"
b) Alamat rumah kita adalah *"address"* atau IP
c) Kotak surat yang ada dirumah kita adalah *"Port"*
d) Kantor pos adalah *"network/* jaringan"
e) *"Socket"* adalah kunci yang memberikan akses untuk menuju *Port* yang tepat/benar

Socket Desriptor

Int fd /* used by accept() */



contoh :

```
int fd;      /* socket descriptor */
if ((fd = socket(AF_INET,
SOCK_STREAM, 0)) < 0) }
      perror("socket");
      exit(1);}
```

Pemrograman *Socket* dengan TCP

Ilustrasi pemrograman *socket* dengan TCP adalah

a) *Client* harus menghubungi *server*
b) Proses pada *server* harus berjalan dahulu pertama kali

c) *Server* harus menciptakan atau membuat *socket* untuk menyambungkan *request*/panggilan *client*
d) *Client* menghubungi *server* dengan:
   - Membuat *server client local* TCP
   - Memberikan alamat IP dan nomor *port* dari proses *server*
e) Ketika *client* menciptakan *socket : client* TCP melakukan koneksi ke *server* TCP
f) Ketika dihubungi oleh *client : Server* TCP membuat *socket* baru untuk proses *server* sehingga bisa berkomunikasi dengan *client*

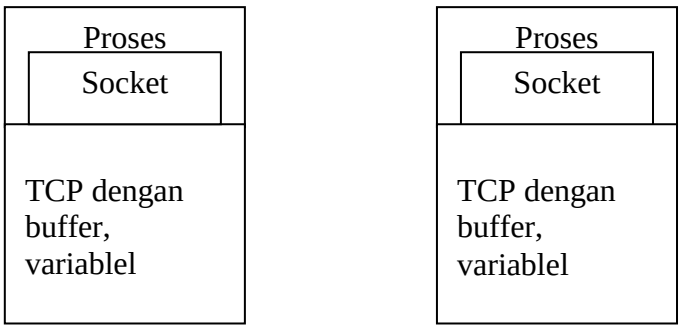
Hal-hal yang harus diperlukan/dibutuhkan untuk komunikasi socket adalah :
a) IP *Address* sumber
b) *Port* sumber
c) IP *Address* tujuan
d) *Port* tujuan

Java Socket

Pada bahasa pemrograman Java, socket dibutuhkan untuk membuat suatu hubungan ke mesin atau proses lain, baru kemudian kita akan mendapatkan *input stream* dan *output stream* untuk pertukaran datanya. Terdapat dua buah *class* yang tersedia pada Java untuk mendukung koneksi dengan tipe *connection-oriented* yaitu :
- java.net.serversocket, digunakan server untuk listen koneksi
- java.net.socket, digunakan client untuk menginisialisasi koneksi

Setelah *client* membentuk suatu koneksi *socket* dengan proses *server*, Server *socket* akan mengembalikan status server ke *client* melalui koneksi yang telah terbentuk sebelumnya. Java juga menyediakan suatu class yang mendukung tipe koneksi *connectionless*, yaitu: "java.net.datagram.socket".
Secara umum pada paket java.net berisi clas class dan *interface* yang menyediakan API *(Application Program Interface)* level terendah (TCP dan UDP) dan level tinggi.

Class Socket TCP



Sumber: www.javaworld.com
Gambar 4. class socket TCP

Untuk class socket TCP:
a) Digunakan oleh *client* dan *server*
b) *Socket* yang baru dibuat dengan menggunakan *constructor socket* ()
c) 4 konstruktor dan 2 *protected*

Untuk *class ServerSocket* TCP

a) Digunakan untuk *server*
b) *Socket* yang baru diciptakan dengan menggunakan *constructor ServerSocket*()
c) 3 *constructor*
d) Gunakan fungsi *accept*() untuk mendapatkan koneksi selanjutnya

Contoh Pemrograman *Socket* dengan TCP

Contoh aplikasi *client server*:

1) *Client* membaca kalimat dari standard *input (inform Userstream)*, mengirim ke server melalui *socket (OutToServerstream)*
2) *Server* membaca kalimat dari *socket*
3) *Server* mengkonversi kalimat menjadi huruf besar, lalu mengirim kembali ke client
4) *Client* membaca, mencetak kalimat yang telah diubah dari socket *(InFromServerStream)*
5) Untuk mengakhiri *socket* bila telah selesai digunakan, *socket* harus ditutup
   a. Status = *close* (s)
   b. Status " 0" jika sukses dan "-1" jika error

Beberapa hal yang perlu diperhatikan untuk membuat aplikasi seperti di atas antara lain adalah:

1. Meng-*import library* yang ada pada Java
   Library java yang harus diimport antara lain adalah :
   Import java.io.*
   Import java.net.*;
2. Kemudian kita membuat *class* untuk *client* dan *class* untuk *server*
3. Untuk *class client*
   a. Buat input stream
      - *bufferedReader inFromUser = new BufferedReader(new nputStreamReader(System.in))*
   b. Buat *socket client* untuk *connect* ke *server*
      - *Socket clientSocket = new socket ("hostname", 6789)*
   c. Buat *output stream* ke *socket*
      - *DataOutputStream out to server=new DataOutputStream(clientSocket.getOutputStream())*
   d. Buat Input stream ke socket
      - *bufferedReader inFromServer=new Buffered Reader(new InputStream Reader(clientSocket.getInput Stream()))*
   e. Kirim kalimat ke server
      $Sentence=inFromUser.readLine()
      OutToServer.writeBytes(sentence+' \n'
   f. Baca kalimat dari server
      - *modifiedSentence= inFromServer.readLine()*
      System.out.println("Dari server :" + modifiedSentence)
      clientSocket.close()
4. Untuk *class server*
   a. Buat *socket Welcome*
      - *ServerSocket Welcome = new ServerSocket(6789))*
   b. Buat *inputstream* ke *socket*
   c. Buat *outputstream* ke *socket*
   d. Baca kalimat dari *socket*
      - *clientSentence= inFromClient.readLine()*
   e. Tulis kalimat ke *socket*
      - *outToClient.writeBytes*

Kesimpulan

Pemahaman protokol-protokol internet dan OSI layer akan sangat membantu kita untuk mengembangkan pemrograman *socket*, karena pemrograman jaringan tidak

hanya sebatas dua atau tiga komputer saja akan tetapi juga jutaan bahkan ratusan juta komputer yang terhubung dan bisa saling berkomunikasi. Mengingat banyaknya komputer yang saling berkomunikasi sudah tentu akan berdampak positif dan negatif. Kebutuhan akan kemanan/ security internet sangat terasa belakangan ini, dengan pemahaman dan pengembangan pemrograman *socket* lebih lanjut tentunya ini merupakan peluang yang menjanjikan.

Daftar Pustaka

http://geeks.netindonesia.net/blogs/agus/default.aspx

http://www.osix.net/modules/article/index.php?id=25

http://peopleenterprise.com/files/9/c_language/default.aspx

http://java.sun.com/docs/books/tutorial/networking/sockets/index.html

http://java.sun.com/developer/onlineTraining/Programming/BasicJava2/socket.html

http://www.javaworld.com/javaworld

http://www.cafeaulait.org/slides/sd2003west/sockets/

http://jan.netcomp.monash.edu.au/distjava/socket/lecture.html

http://www.awprofessional.com/articles/article.asp?p=27633&seqNum=5&rl=1