

Implementasi Java Pada Pemantauan Data Hasil Akuisisi Secara *Real Time* Melalui Internet

Mushlihudin

mdin@elektrouad.net, mdin@grahadesign.com
<http://udin.te.uad.ac.id>

Lisensi Dokumen:

Copyright © 2006 IlmuKomputer.Com

Seluruh dokumen di **IlmuKomputer.Com** dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.

Abstrak

Pekerjaan atau penelitian yang memerlukan pemantauan secara langsung dan dapat dilakukan setiap saat dari tempat yang berjauhan dengan obyek menjadi kebutuhan pada era informasi. Apalagi proses-proses yang bisa diamati oleh beberapa orang secara bersamaan dari tempat yang berbeda-beda secara aman dan dapat mendukung dalam pengambilan keputusan, merupakan sesuatu yang diharapkan oleh berbagai pihak.

Penelitian ini mengkaji metode untuk membaca data hasil pengukuran software akuisisi menggunakan internet. Sebuah sistem aplikasi yang dibuat menggunakan bahasa Java dengan arsitektur multithreading dan client-server yang diterapkan pada protokol TCP/IP memudahkan pengguna mengakses data tersebut dari manapun. Sebuah applet yang ditempelkan dalam dokumen HTML dapat diakses oleh browser, mengurangi biaya pengadaan protokol dan software untuk pengamatan. Penelitian ini mengkaji pada aspek fungsi dan cara melayani banyak pengguna yang mengakses secara bersamaan.

Berdasarkan pengujian menunjukan bahwa arsitektur jaringan, waktu akses, dan jumlah pengguna berpengaruh pada aspek fungsi. Akses yang dilakukan pada jam sibuk, jarak pengguna dengan server dan jumlah pengguna yang lebih banyak berdampak pada fungsi yang tidak dapat bekerja secara maksimal.

Keyword : pemantauan, real time, java, tcp/ip, proses

1. Pendahuluan

Pekerjaan atau penelitian yang memerlukan pemantauan secara langsung dan dapat dilakukan setiap saat dari tempat yang berjauhan dengan obyek menjadi kebutuhan pada era informasi. Banyak pihak yang berharap terhadap pemantauan proses yang bisa dilakukan oleh beberapa orang secara bersamaan namun dari tempat yang berbeda-beda dan aman.

Pengamatan dengan memanfaatkan jaringan komputer merupakan salah satu alternatif yang dapat digunakan untuk pemantauan proses-proses secara *real time*. Jangkauan jaringan komputer melingkupi ke seluruh belahan bumi memberikan kesempatan secara leluasa.

Penelitian ini bertujuan untuk mengkaji penggunaan Pemrograman Java pada internet dan menampilkan data secara visual. Penggunaan internet sebagai media transmisi data mengurangi biaya pengadaan infrastruktur jaringan yang mahal.. Disamping itu memberikan keleluasaan pengguna dalam menggunakannya karena tidak perlu menggunakan software secara khusus. Namun cukup menggunakan *browser* yang sudah umum dipakai dalam akses internet.

2. Model Jaringan Komputer

Dilihat dari sistem layanan, ada dua model jaringan komputer yaitu:

1. **Model Peer to Peer.** Pada model ini setiap komputer dalam jaringan dapat menawarkan dan mengambil layanan dari *peer* lain. Model ini cocok untuk jaringan kecil.
2. **Model Client-Server.** Pada model ini hanya komputer server yang dapat memberikan layanan, sedangkan pada komputer *client* hanya dapat menerima layanan. Pada jaringan model ini, beberapa komputer dapat di set sebagai server yang dapat memberikan layanan pada komputer lain sebagai *client* yang terhubung ke jaringan.

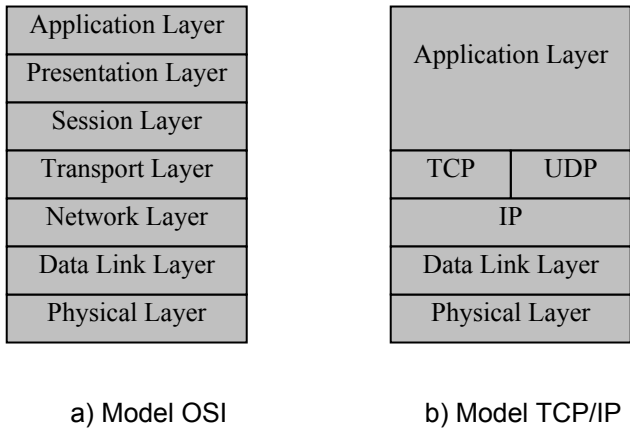
3. TCP/IP

Model referensi adalah model yang digunakan untuk menjelaskan bagaimana komponen dari satu sistem dapat saling berinteraksi satu sama lain dan menjelaskan bagaimana spesifikasi antar muka yang dapat digunakan diantara dua komponen [4]. Dua arsitektur jaringan yang penting adalah model referensi OSI (*Open System Interconnection*) dan model referensi TCP/IP [5].

Model referensi TCP/IP pada awalnya dibuat untuk memenuhi kebutuhan dalam komunitas akademik dan bidang pertahanan, sedangkan model referensi OSI diciptakan berdasarkan sebuah proposal yang dibuat oleh *International Standards Organization* (ISO) sebagai awal standarisasi protokol internasional yang digunakan pada berbagai lapisan. Model ini juga disebut *ISO OSI Reference Model* karena model ini ditujukan bagi pengembangan sistem terbuka (*open system*). Model referensi TCP/IP dan OSI dapat dilihat pada Gambar 1.

TCP/IP merupakan protokol yang banyak dipakai pada jaringan komputer, karena mempunyai kelebihan-kelebihan sebagai berikut [2]:

1. merupakan protokol standar yang terbuka, tidak tergantung terhadap perangkat keras dan sistem operasi.
2. tidak tergantung pada merk perangkat keras jaringan maupun perangkat lunak tertentu, sehingga mampu dan cocok untuk diintegrasikan dan diimplementasikan pada berbagai jaringan komunikasi dan media transmisi.
3. mempunyai model pengalamatan global yang unik, memungkinkan perangkat IP mengidentifikasi dan diidentifikasi pada jaringan global.
4. distandarisasi dalam bentuk RFC (*Request For Comment*) yang dipublikasikan dan dapat diperoleh secara gratis.
5. memiliki fasilitas routing yang dapat diterapkan pada *internetworking*.
6. mampu menyediakan layanan yang luas pada pengguna, karena distandarkan sebagai protokol level tinggi.



Gambar 1. Model OSI dan TCP/IP [3]

4. Bahasa Pemrograman Java

Java merupakan bahasa pemrograman tingkat tinggi yang dirancang untuk dijalankan pada sistem jaringan. Java dikembangkan oleh perusahaan SUN Microsystems yang dimotori oleh James Gosling, seorang anggota kehormatan SUN dan pakar komputer yang jenius.

Berdasarkan *white paper* yang ditulis oleh perancang bahasa pemrograman Java, Java memiliki keunggulan-keunggulan sebagai berikut [6]:

1. **Sederhana.** Bahasa pemrograman Java sangat sederhana sebagai suatu bahasa pemrograman yang berorientasi obyek.
2. **Berorientasi Obyek (*Object Oriented*).** Java merupakan bahasa pemrograman yang berorientasi pada obyek. Rancangan berorientasi obyek merupakan suatu teknik yang memusatkan rancangan pada data (obyek) dan antar muka.
3. **Terdistribusi.** Java memiliki *library* rutin yang lengkap untuk dirangkai pada protokol TCP/IP dengan mudah, seperti HTTP dan FTP. Kemampuan *networking* Java lebih kuat dan mudah dipakai.
4. **Kuat (*Robust*).** Suatu program yang dibuat dengan Java dapat dipercaya dalam berbagai hal, karena Java banyak menekankan pada pengecekan pada saat *run time* dan mengurangi kemungkinan timbulnya kesalahan (*error*)..
5. **Aman (*Secure*).** Java dimaksudkan pada pemrograman jaringan dan sistem distribusi sehingga penekanan ditujukan pada masalah keamanan. Java memungkinkan untuk membuat suatu program yang bebas virus dan sistem yang bebas dari kerusakan, karena Java membuat suatu sistem yang memiliki mekanisme keamanan sistem yang kuat.
6. **Netral Arsitektur.** Java dirancang untuk mendukung aplikasi yang beroperasi di lingkungan jaringan yang heterogen, sehingga aplikasi tersebut harus dapat bekerja pada bermacam-macam arsitektur perangkat keras dan sistem operasi serta harus mampu bekerja sama dengan berbagai antarmuka bahasa pemrograman. Java dapat berjalan pada *multiplatform*; Windows, Linux, Solaris, dan Mac OS
7. **Portable.** Spesifikasi Java tidak bergantung pada lingkungan implementasi.
8. **Interpreter.** *Interpreter* Java dapat mengeksekusi kode byte Java secara langsung pada setiap mesin yang terdapat interpreter dan sistem *run-time* Java. Pada sistem Java, tahap *link* dari program adalah sederhana, bertahap dan ringan. Hal ini membuat siklus pengembangan menjadi sangat cepat.

- 9. **Kinerja yang tinggi.** Java dapat mencapai performansi yang tinggi dengan cara mengadopsi sebuah skema yang memungkinkan interpreter dapat berjalan pada kecepatan penuh tanpa perlu memeriksa lingkungan *run-time*.
- 10. **Multithreading.** *Multithreading* adalah kemampuan suatu program komputer untuk melakukan beberapa pekerjaan sekaligus, misalnya mencetak *file* sambil *browsing* ke internet.
- 11. **Dinamis.** Java dirancang untuk beradaptasi dengan lingkungan yang sedang berkembang. Walaupun *compiler* Java ketat dalam proses kompilasinya, namun bahasa dan sistem *run-time* Java dinamis dalam tahap *linking*-nya.

Java mampu memenuhi kebutuhan sistem yang terhubung ke jaringan, berorientasi objek, *multilink* dan *multiflatform*. Komponen dalam Java *network programming* adalah *stream*, *socket* dan *port*, serta *threads* [1].

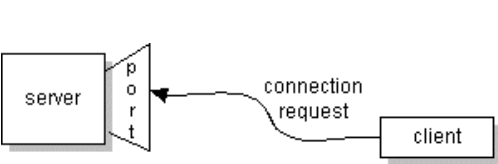
5. **Socket dan Port**

Secara umum, sebuah server yang dijalankan pada sebuah komputer memiliki *socket* yang terhubung dengan nomor *port* tertentu. Server senantiasa menunggu datangnya *client* yang menghubunginya melalui *socket*. Jika ada *client* yang masuk melalui *socket* selanjutnya server akan memberikan tanggapan. Pada Java sebuah *socket* mewakili koneksi TCP, dengan menggunakan *class* ini maka sebuah *client* dapat membuat komunikasi *stream* dengan komputer lain.

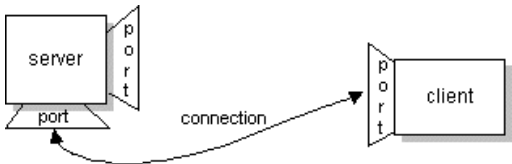
Untuk berkomunikasi dengan komputer lain menggunakan TCP/IP, *client* pertama kali membuat *socket* menuju lawannya, hal ini otomatis membuat hubungan TCP. *Socket* digunakan untuk menentukan nama *host* dari server sekaligus nomor *port* TCP yang digunakan.

Port berjumlah 65.536 (2¹⁶) yang merupakan bilangan integer berharga dari 1 sampai dengan 65.535. Sebuah *client* yang menghubungi server harus menentukan nama server dan nomor *port*, seperti dalam gambar 2.a. Sebuah server yang secara aktif mendengar pada *port* tertentu yang telah dibukanya menerima hubungan tersebut.

Setelah sebuah *socket* dibuat, maka *method socket* akan memperoleh *stream* sehingga *client* dan server dapat berkomunikasi, seperti pada gambar 2.b.



Gambar 2a. *Client* menghubungi Server



Gambar 2b. Koneksi *client* dan server

Pada penelitian ini, *ServerData* yang memiliki fungsi untuk melayani permintaan dari *client* menggunakan *Socket* dengan *port* 8888, yang dapat dilihat *source codenya* pada gambar 3.

```
public class ServerData {  
    public static void main (String args[]) throws IOException {  
        int port = 8888; //Integer.parseInt (args[0]);  
        ServerSocket server = new ServerSocket (port);  
        while (true) {
```

```
        System.out.println ("Server Running.... ");
        Socket client = server.accept ();
        System.out.println ("Accepted from " + client.getInetAddress
    ());
        ClientHandler handler = new ClientHandler (client);
        handler.start ();
    }
}
}
```

Gambar 3. ServerData Menggunakan Socket dengan port 8888.

6. Arsitektur jaringan sistem pemantauan

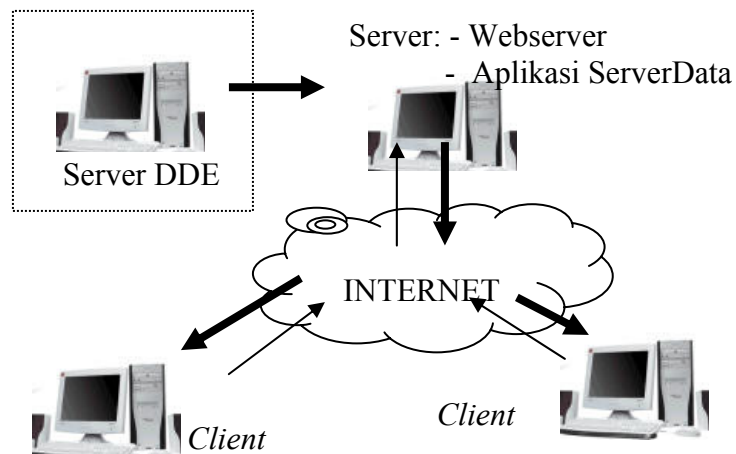
Sistem yang dirancang pada penelitian ini menggunakan model *client-server* seperti dapat dilihat pada gambar 4. Dalam arsitektur ini, terdapat dua server yaitu:

- 1. Server DDE yang bertugas membaca data dari komputer akuisisi untuk dikirimkan ke ServerData dan
- 2. ServerData yang bertugas mendistribusikan dan melayani permintaan *client* yang akan memantau proses.

Secara jelas mengenai jenis komponen dan tugasnya dapat juga dilihat pada Tabel 1.

Tabel 1. Jenis komponen dari Sistem Pemantauan

Komponen	Tugas
1. Server DDE	Menangkap data dari Komputer Akuisisi dan mengirimkannya ke ServerData
2. ServerData	Menampung data dari DDEServer dan mengirimkannya ke <i>client</i> Menunggu koneksi dari <i>client</i> dan menanganinya dengan membuatkan <i>thread</i> baru.
3. <i>Client</i>	Memantau proses dengan menghubungi ServerData



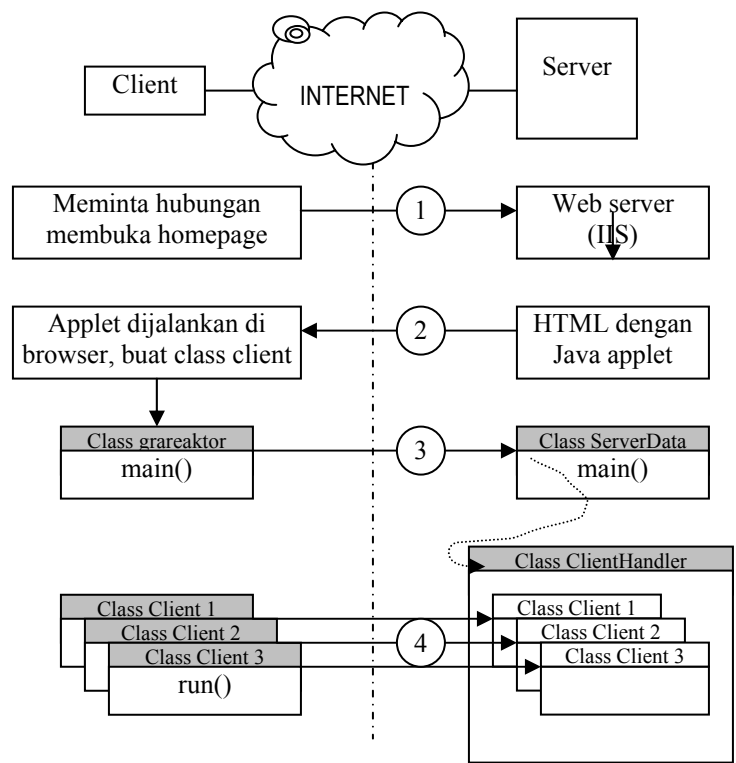
Gambar 4. Arsitektur rancangan sistem pemantauan

7. Penanganan *Client* dengan *Multithreading*

Secara tradisional, aliran eksekusi sebuah program hanya ada satu aliran Eksekusi dimulai dari awal program kemudian mengalir melalui jalur tunggal sampai program selesai. Java mengatasi cara konvensional itu dengan metode *Threading*.

Thread memungkinkan eksekusi sebuah program secara efisien dan membuat server yang kokoh karena setiap *client* dipisahkan pada *thread* yang berbeda. Jika ada banyak *client* maka teknik *multithreading* digunakan untuk menanganinya, sehingga terdapat banyak aliran, atau *threads*, yang dieksekusi dalam program yang sama pada saat bersamaan.

Pada penelitian ini, penanganan permintaan dari client oleh server dengan teknik *multithreading* dapat dilihat dalam gambar 5, dan *source code* penanganannya dapat dilihat pada gambar 6.



Gambar 5. Penanganan Client dengan Multithreading

```
while (enum.hasMoreElements ()) {
    ChatHandler handler =(ChatHandler) enum.nextElement ();
    try {
        handler.dataOut.writeUTF(message);
        handler.dataOut.flush ();
    } catch (IOException ex) {
        handler.stop ();
    }
}
```

Gambar 6. Proses pengiriman data ke semua client

8. Client menghubungi ServerData dengan Applet

Aplikasi di Client dibuat dengan Java berbentuk *applet* yang berjalan di atas *browser* yang mendukung aplikasi java dapat dieksekusi. Agar ukuran *Applet* lebih kecil dan ringan untuk *download* maka *Applet* dibuat dalam bentuk File JAR.

Setelah *applet* di-*download* dari *server*, *Applet* dieksekusi oleh *client* membuat *class client* baru yang berkomunikasi dengan *server*, *source code* awal aksi ini dapat dilihat pada gambar 7.

Proses kerja *applet* sejak di-*load* sampai di-*close* dapat diterangkan sebagai berikut :

1. Inisialisasi terjadi saat *applet* di-*load* pertama kali. Pada tahap ini ditentukan nilai awal variabel, status awal *applet*, dan pembacaan parameter.
2. Setelah *applet* selesai di-*download*, *Run* eksekusi *applet* dengan membuat koneksi ke *ServerData* yaitu membuat *socket* baru dengan informasi *host* dan *port server*, sedangkan *server* akan menerimanya sebagai permintaan koneksi.
3. Setelah *client* mendapat koneksi dari *server* kemudian dibuat *input-output stream* yang berhubungan dengan satu *handler* di *server*.
4. Data yang diterima dari *ServerData* divisualkan dalam bentuk trending garis.

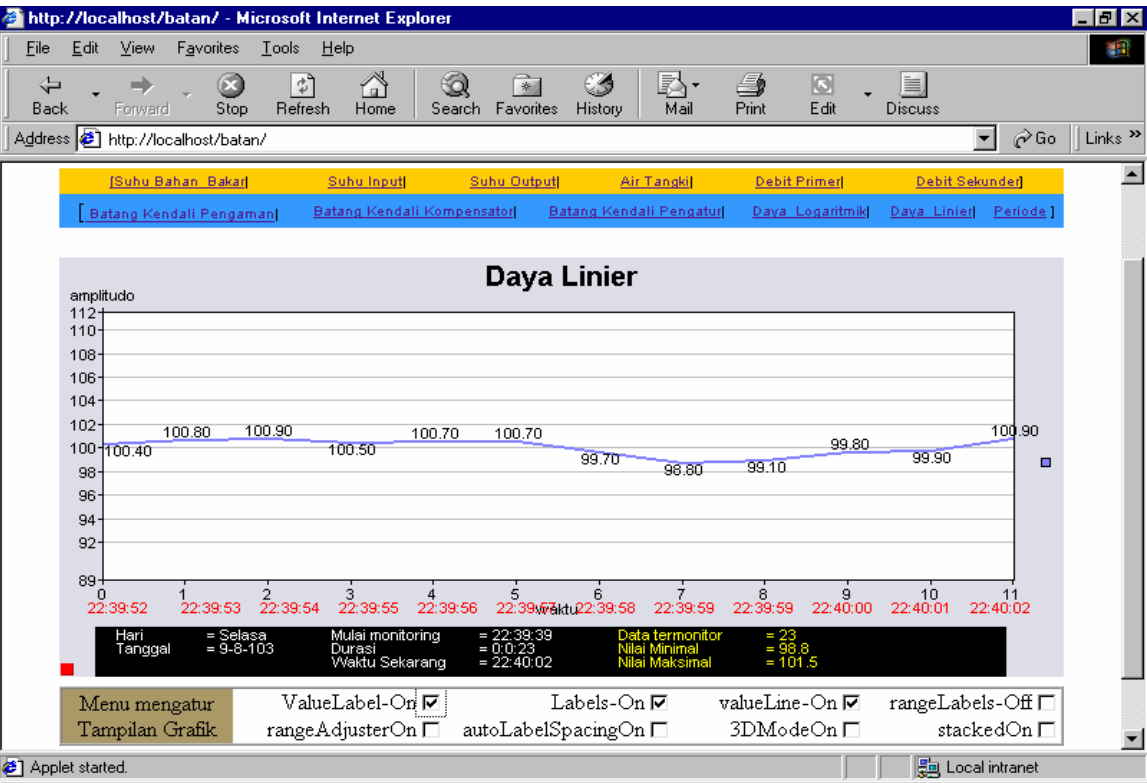
```
try {  
    socket = new Socket("localhost",8009);  
    fromServer = new BufferedReader  
        (new InputStreamReader (socket.getInputStream ()));  
  
    } catch (UnknownHostException e) {  
        System.out.println("Unknown host");  
    } catch (IOException e) {  
        System.out.println("IO Exception");  
        return;  
    }  
}
```

Gambar 7. *Client* Memulai koneksi ke *ServerData*

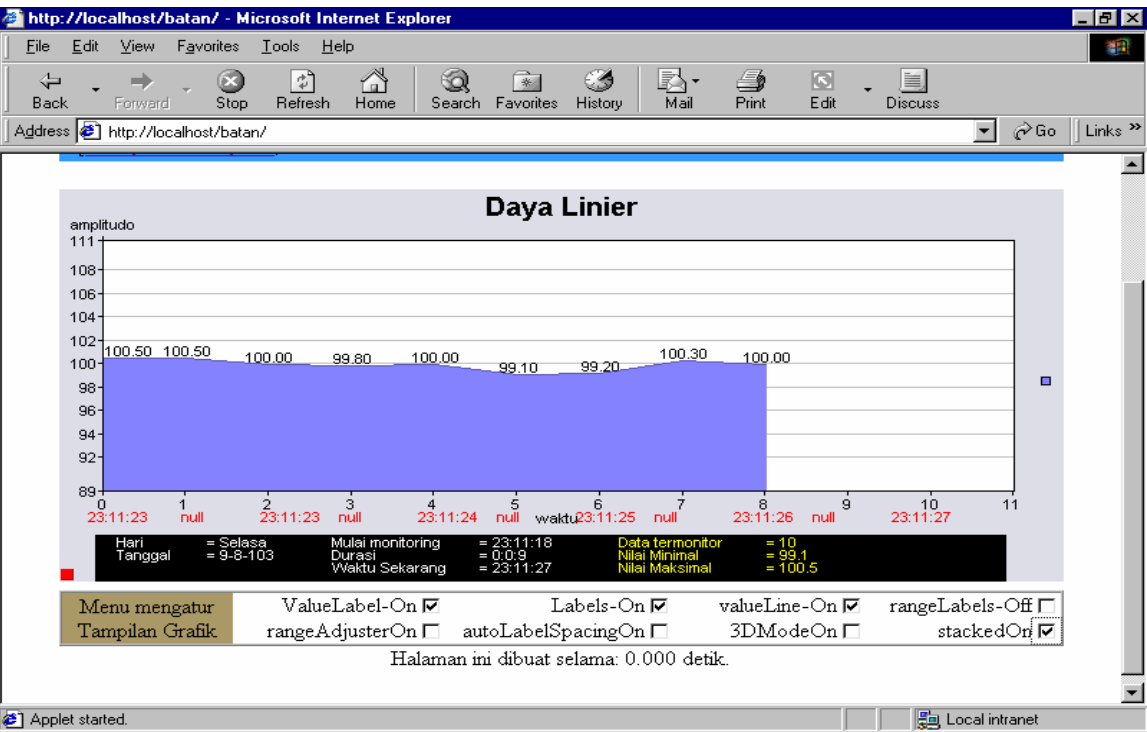
9. Memantau hasil akuisisi dari Sumber Proses

Memantau proses sistem dilakukan dengan menghubungi alamat *URL server*. Setiap pengguna yang akan memantau diminta *login* terlebih dahulu. Fungsi *login* digunakan untuk memantau dan membatasi pengguna yang aktif.

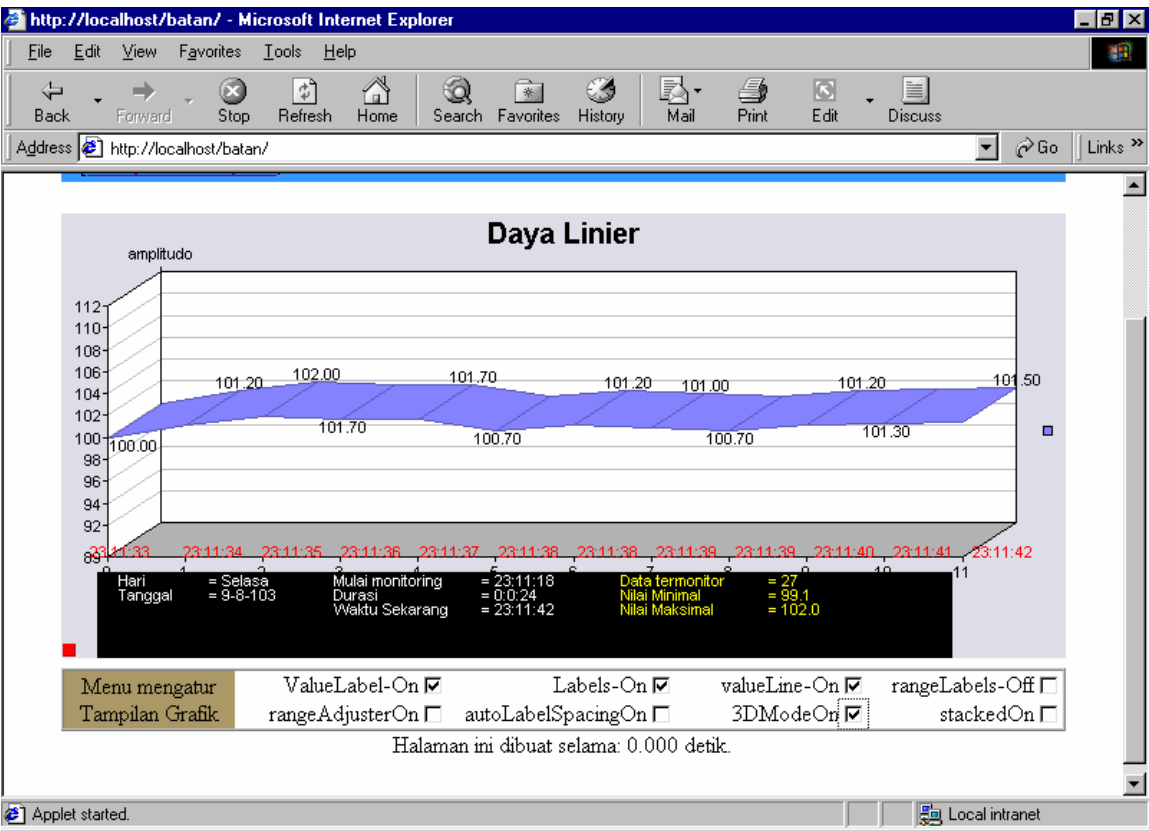
Pengguna sebelum melakukan login dapat mendaftarkan diri secara langsung ke admin. Selama memantau pengguna dapat memilih tipe tampilan yang diinginkan dalam bentuk garis (gambar 8.a), bentuk pancang (gambar 8.b.) dan bentuk 3 dimensi (gambar 8.c).



Gambar 8.a Grafik *trending monitoring* dalam bentuk garis



Gambar 8.b. Grafik *trending monitoring* dalam bentuk pancang



Gambr 8.c Grafik *trending monitoring* dalam bentuk 3 dimensi.

10. Kesimpulan

Java dapat diimplementasikan pada protokol TCP/IP untuk pemantauan proses secara *real time* dan mudah dalam menangani banyak pengguna yang akses secara bersamaan. Dengan arsitektur jaringan yang memisahkan antara sumber data (data dari *plant*) dengan serverdata (data untuk melayani permintaan pengguna) memberikan pengamanan terhadap sumber data.

Pengembangan lain dapat dilakukan dengan menyimpan setiap hasil data hasil akuisisi disimpan dalam database. Manfaat penyimpanan tersebut dapat digunakan untuk keperluan analisa para pengguna terhadap proses yang berlangsung sebelumnya.

Referensi

[1] Huges, M., Shoffner, M., Hammer, D., 1999, **Java Network Programming**, 2nd Edition, Manning Publication Co.,.

[2] Stalling, W., 1997, **Data and Computer Communication**, 5th Edition, Prentice-Hall International Edition.

[3] Feit, S., 1997, **TCP/IP : Architecture, Protocols, and Implementation with Ipv6 and IP Security**, 2nd Edition, McGraw-Hill, New York.

[4] Tung, K.Y., 1997, **Teknologi Jaringan Internet**, Penerbit Andi, Yogyakarta.

[5] Tanenbaum, A.S., 2000, **Jaringan Komputer**, Edisi 3, Jilid 1, Prenhallindo, Jakarta

- [6] Utdirartatmo, F., 2003, **Aplikasi Database di Java dengan JBuilder**, Elex Media Komputindo, Jakarta.

RIWAYAT PENULIS



Mushlihudin lahir di Gunungkidul pada 6 Januari 1967. Setelah lulus dari SMA N 2 Wonosari, Yogyakarta, melanjutkan pendidikan S1 di Jurusan Teknik Elektro Universitas Gadjah Mada (UGM), dengan Judul Skripsi: ***Simulasi ARQ metode stop and wait***, tamat pada tahun 1993 dan menamatkan pendidikan Magister Teknik (S2) Program Studi Teknik Elektro Institut Teknologi Bandung (ITB) tahun 2003, dengan judul tesis: ***Monitoring Proses Iradiasi Reaktor Nuklir melalui Internet***. Saat ini sebagai staf edukatif di Program Studi Teknik Elektro Universitas Ahmad Dahlan (UAD) Yogyakarta dan Kepala Pusat Studi dan Pelayanan bidang Teknologi Informasi UAD. Aktif dalam Computer Network and Information Technology Study Center (CoNIT) Teknik Elektro UAD. Menjadi anggota Redaksi Jurnal Ilmiah Teknik Elektro "Telkomnika" sejak 2003. Terlibat dan memotori lahirnya Kelompok Studi Linux UAD tahun 1998. Kini juga menjadi konsultan dalam Implementasi Teknologi Informasi untuk Perusahaan dan Pendidikan, dan sejak 2005 hingga kini menjadi Anggota TIM Jogja E-Education. Telah berhasil melakukan restrukturisasi dan pengembangan Jaringan Terpadu UAD dengan jangkauan Metropolitan Area Network. Pernah mengikuti training Linux Full Package dan Security Linux: fundamental and advanced, serta pengendalian sistem berbasis PLC. Aktif dalam memberikan training-training untuk dosen dan karyawan dalam bidang teknologi informasi, internet dan aplikasinya. Obsesi yang sedang dikembangkan yaitu membangun layanan administrasi Kantor, Perguruan Tinggi dan Perusahaan berbasis linux. Bidang yang ditekuni yaitu, sekuriti teknologi informasi, E-Commerce, dan perancangan sistem dan teknologi informasi.