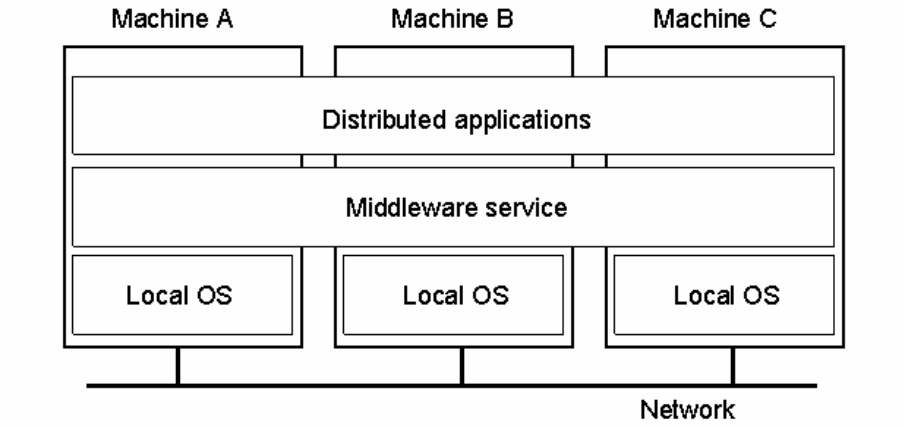
Aturan Praktikum:

- Keterlambatan maksimal 10 menit
- Hasil praktikum, selesai atau tidak selesai ditanggung mahasiswa
- Nilai praktikum adalah nilai perorangan, sebagai tambahan nilai test atau quiz jika nilai test atau quiz kurang mencukupi
- Mahasiswa login ke komputer
- Persiapan dan penjelasan praktikum 10 menit
- Praktikum dilakukan selama 70 menit terdiri dari praktikum sesuai modul
- Hasil praktikum di-zip dan dikumpulkan
- Sisa waktu 20 menit digunakan untuk tes kecil hasil praktikum, saat tes tampilan komputer adalah background desktop dan mahasiswa tidak diperkenankan menggunakan komputer dan melihat modul praktikum
- Mahasiswa logout dari komputer
- Praktikum selesai

PENGANTAR APLIKASI TERDISTRIBUSI
(Minggu I – Praktikum I)

A. SEKILAS APLIKASI TERDISTRIBUSI

Sistem terdistribusi adalah kumpulan komputer yang berdiri sendiri yang dapat dilihat oleh pemakainya sebagai satu kesatuan sistem.



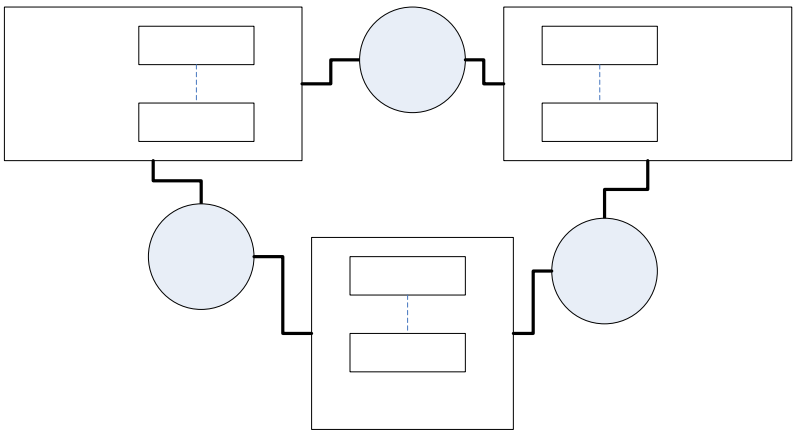
Aplikasi terdistribusi merupakan bagian dari sistem terdistribusi. Aplikasi terdistribusi adalah aplikasi yang dibuat di atas komponen yang berbeda dan dijalankan pada lingkungan yang berbeda dan biasanya terhubung dengan menggunakan jaringan. Aplikasi terdistribusi biasanya berbasis pada arsitektur client-server.

Komputasi terdistribusi adalah proses berjalannya sebuah aktifitas komputasi yang dilakukan lebih dari satu komputer. Aplikasi yang berjalan pada komputasi

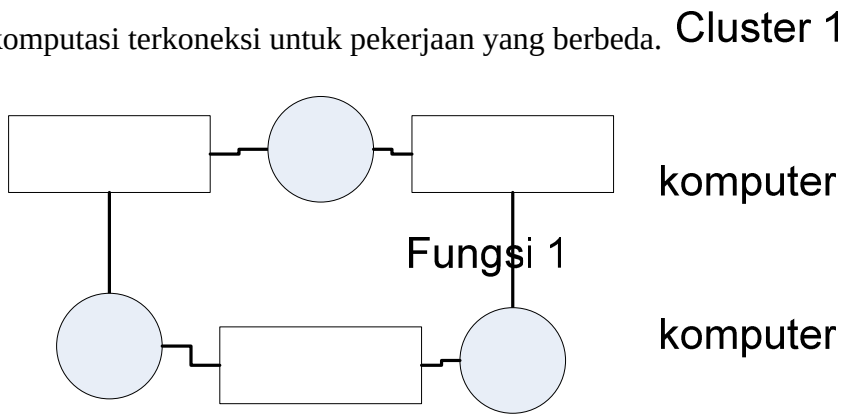
terdistribusi inilah yang disebut sebagai aplikasi terdistribusi. Arsitektur komputasi terdistribusi yaitu client-server dan peer to peer.

Berikut dua buah cara komputasi yang perlu diketahui:

Cluster Computing: komputasi terkelompok dengan fungsi terkelompok dan bekerja bersamaan.



Grid Computing: komputasi terkoneksi untuk pekerjaan yang berbeda.

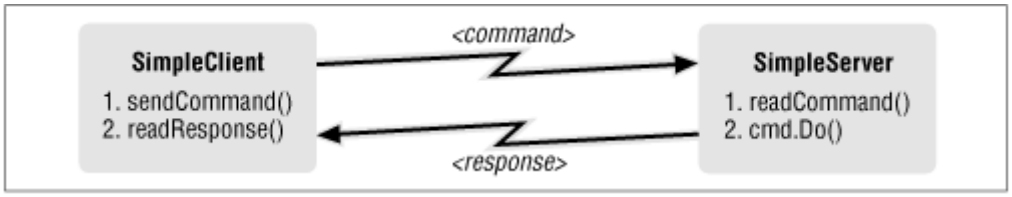


B. CLIENT SERVER

Client-Server adalah salah satu arsitektur aplikasi terdistribusi dimana client bertindak sebagai pemakai layanan (*service*) dan server sebagai penyedia layanan (*service*) yang harus terus ada untuk memenuhi permintaan (*request*) layanan dari client. Server diakses client dengan alamat tertentu.

1. Praktikum

Membuat program simulasi client-server dengan arsitektur sebagai berikut:



a. Persiapan

- Membuat direktori kerja dengan nama SI319-P1-Kelas-NIM misalnya SI319-P1-A-23507024
- Di dalam direktori di atas, buat direktori ClientServer untuk menyimpan file-file yang akan dibuat

b. File-file yang harus dibuat

Nama File: SimpleServer.java

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

public class SimpleServer {
    public static void main(String[] args) throws IOException {

        ServerSocket serverSocket = null;

            // mengecek jumlah masukan
            if (args.length < 1) {
            System.out.println("Usage: java SimpleServer [port]");
            System.exit(1);
          }

            // mengecek masukan port
          int port = 3000;
          try {
            port = Integer.parseInt(args[0]);
          }catch (NumberFormatException e) {
          }
```

```
            // membuka koneksi socket
        try {
            serverSocket = new ServerSocket(port);
        } catch (IOException e) {
            System.err.println("Could not listen on port: " + port
              + ".");
            System.exit(1);
        }

            // menunggu koneksi klien
        Socket clientSocket = null;
        try {
            clientSocket = serverSocket.accept();
        } catch (IOException e) {
            System.err.println("Accept failed.");
            System.exit(1);
        }

            // komunikasi dengan klien dengan protokol
        PrintWriter out =
            new PrintWriter(clientSocket.getOutputStream(), true);
        BufferedReader in = new BufferedReader(
                        new InputStreamReader(
                        clientSocket.getInputStream()));
        String inputLine, outputLine;
        SimpleProtocol kkp = new SimpleProtocol();

        while ((inputLine = in.readLine()) != null) {
             outputLine = kkp.processInput(inputLine);
             out.println(outputLine);
             if (outputLine.equals("Bye."))
                break;
        }

            // koneksi ditutup
        out.close();
        in.close();
        clientSocket.close();
        serverSocket.close();
    }
}
```

Nama File: SimpleProtocol.java

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

public class SimpleProtocol {

    public String processInput(String theInput) {
            // memproses kata balikan dari server ke klien
        String theOutput = null;
            theOutput = "Server: " + theInput;
        return theOutput;
    }
}
```

Nama File: SimpleClient.java

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

public class SimpleClient {
    public static void main(String[] args) throws IOException {

            // mengecek jumlah masukan
        if (args.length < 2) {
          System.out.println("Usage: java SimpleClient [host] [port]");
          System.exit(1);
        }

            // mengecek masukan
        String host = args[0];
        int port = 3000;
        try {
            port = Integer.parseInt(args[1]);
        }catch (NumberFormatException e) {
        }

        Socket echoSocket = null;
        PrintWriter out = null;
        BufferedReader in = null;

            // membuka koneksi socket
        try {
            echoSocket = new Socket(args[0], port);
            out = new PrintWriter(echoSocket.getOutputStream(), true);
            in = new BufferedReader(new InputStreamReader(
                                        echoSocket.getInputStream()));
        } catch (UnknownHostException e) {
            System.err.println("Don't know about host: " + args[0]
              + ".");
            System.exit(1);
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for "
                               + "the connection to: " + args[0]
                               + ".");
            System.exit(1);
        }

            // komunikasi dengan server
        BufferedReader stdIn = new BufferedReader(
            new InputStreamReader(System.in));
        String userInput;

        while ((userInput = stdIn.readLine()) != null) {
            out.println(userInput);
            System.out.println("from client in " + in.readLine());
        }

            // koneksi ditutup
        out.close();
```

```
        in.close();
        stdIn.close();
        echoSocket.close();
    }
}
```

c. Menjalankan aplikasi client server sederhana

1. Buka Command Prompt, masuk ke direktori ClientServer
2. ketik: javac
   jika muncul pesan:

```
'javac' is not recognized as an internal or external command,
operable program or batch file.
```

Maka ketik perintah:

path=<tempat direktori instalasi java>\bin;%path%

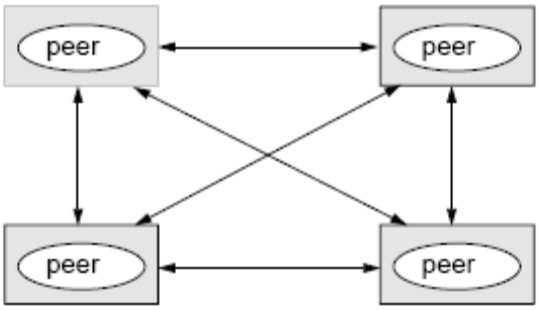
contoh:

path= C:\Program Files\Java\jdk1.5.0_05\bin;%path%

3. Ketik perintah: javac *.java
4. Jika muncul *error* perbaiki kode program sesuai dengan pesan kesalahan yang muncul dan kompilasi ulang dengan perintah javac
5. Jalankan aplikasi dengan mengetik perintah: java SimpleServer [port] misalnya
   java SimpleServer 3000
6. Buka Command Prompt, masuk ke direktori ClientServer, ulangi langkah no. 2
7. Ketik perintah: java SimpleClient [host] [port] misalnya
   java SimpleClient localhost 3000
8. Tuliskan apapun pada jendela client untuk berkomunikasi dengan server
9. Untuk menghentikan program ketik: ctrl+c pada jendela client

C. PEER TO PEER

Peer to peer adalah arsitektur aplikasi terdistribusi dimana setiap mesin yang terhubung dapat memberikan atau meminta layanan (*service*) ke mesin lain. Peer to peer biasanya digunakan untuk *resource sharing* atau lebih sering digunakan untuk *file sharing*. Alamat mesin pada peer to peer bukan sebuah alamat yang fix. Koneksi tidak stabil antar mesin karena mesin yang tergabung dalam peer to peer dapat terhubung maupun meninggalkan koneksi tanpa ada batasan.



D. CLUSTERING

Sistem kluster ialah gabungan dari beberapa sistem individual (komputer) yang dikumpulkan pada suatu lokasi, saling berbagi tempat penyimpanan data (*storage*), dan saling terhubung dalam jaringan lokal (*Local Area Network*). Sistem kluster memiliki menggabungkan beberapa CPU untuk meningkatkan kinerja komputasi. Jika salah satu mesin mengalami masalah dalam menjalankan tugas maka mesin lain dapat mengambil alih pelaksanaan tugas itu. Dengan demikian, sistem akan lebih andal dan *fault tolerant* dalam melakukan komputasi.

Dalam hal jaringan, sistem kluster mirip dengan sistem terdistribusi (*distributed system*). Bedanya, jika jaringan pada sistem terdistribusi melingkupi komputer-komputer yang lokasinya tersebar maka jaringan pada sistem kluster menghubungkan banyak komputer yang dikumpulkan dalam satu tempat.

Isu yang menarik tentang sistem kluster ialah bagaimana mengatur mesin-mesin penyusun sistem dalam berbagi tempat penyimpanan data (*storage*). Untuk saat ini, biasanya sistem kluster hanya terdiri dari dua hingga empat mesin berhubung kerumitan dalam mengatur akses mesin-mesin ini ke tempat penyimpanan data.

Isu di atas juga berkembang menjadi bagaimana menerapkan sistem kluster secara paralel atau dalam jaringan yang lebih luas (*Wide Area Network*). Hal penting yang berkaitan dengan penerapan sistem kluster secara paralel ialah kemampuan mesin-mesin penyusun sistem untuk mengakses data di *storage* secara serentak. Berbagai *software* khusus dikembangkan untuk mendukung kemampuan itu karena kebanyakan sistem operasi tidak menyediakan fasilitas yang memadai. Salah satu contoh perangkat-lunak-nya-nya ialah *Oracle Parallel Server* yang khusus didesain untuk sistem kluster paralel.

Seiring dengan perkembangan pesat teknologi kluster, sistim kluster diharapkan tidak lagi terbatas pada sekumpulan mesin pada satu lokasi yang terhubung dalam jaringan lokal. Riset dan penelitian sedang dilakukan agar pada suatu saat sistem kluster dapat melingkupi berbagai mesin yang tersebar di seluruh belahan dunia.

Komputasi model terbaru ini juga berbasis jaringan dengan *clustered system*. Digunakan *super computer* untuk melakukan komputasinya. Pada model ini komputasi

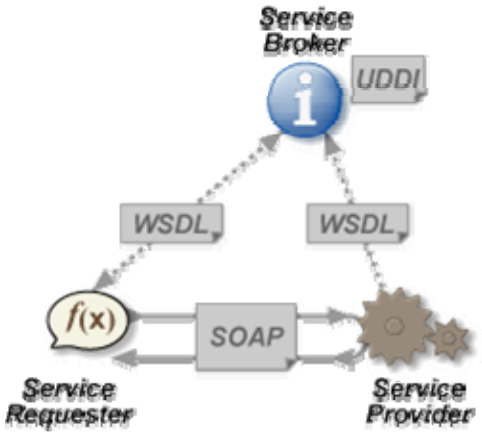
dikembangkan melalui *pc-farm*. Perbedaan yang nyata dengan komputasi berbasis jaringan ialah bahwa komputasi berbasis *grid* dilakukan bersama-sama seperti sebuah *multiprocessor* dan tidak hanya melakukan pertukaran data seperti pada komputasi berbasis jaringan.

E. WEB SERVICES

Web Service merupakan perangkat lunak yang didesain untuk mendukung interoperabilitas (kemampuan bekerjasama) antar mesin yang terhubung dengan jaringan. Web service sebenarnya mirip dengan arsitektur client-server yang berkomunikasi menggunakan XML (*Extensible Markup Language*) yang berupa SOAP (*Simple Object Access Protocol*).

Web services merupakan aplikasi enterprise (kompleks) berbasis web dan dapat digunakan oleh umum. Pustaka Java yang biasa digunakan untuk web services adalah JAX-RPC (*Java API for XML Remote Procedure Calls*) yang didalamnya berisi pustaka untuk keperluan komunikasi dengan SOAP/XML.

Fungsi *web service* adalah menjadi antarmuka yang mendeskripsikan kumpulan operasi yang dapat diakses via jaringan (*network*) dengan menggunakan pesan yang memakai standar XML.



Keterangan gambar:

SOAP (Simple Object Access Protocol)
Berbasis XML, format pesan yang dipakai oleh protokol seperti HTTP dan HTTPS. SOAP adalah standar untuk bertukar pesan-pesan berbasis XML melalui jaringan komputer atau sebuah jalan untuk program yang berjalan pada suatu sistem operasi (OS) untuk berkomunikasi dengan program pada OS yang sama maupun berbeda dengan menggunakan HTTP dan XML sebagai mekanisme untuk pertukaran data. Extensible

Markup Language (XML) adalah bahasa markup serbaguna yang direkomendasikan W3C untuk mendeskripsikan berbagai macam data. XML menggunakan markup tags seperti halnya HTML namun penggunaannya tidak terbatas pada tampilan halaman web saja.

SOAP menspesifikasikan secara jelas bagaimana cara untuk meng-encode header HHTP dan file XML sehingga program pada suatu komputer dapat memanggil program pada komputer lain dan mengirimkan informasi, dan bagaimana program yang dipanggil memberikan tanggapan.

SOAP adalah protokol ringan yang ditujukan untuk pertukaran informasi struktur pada lingkup desentralisasi, dan terdistribusi. SOAP menggunakan teknologi XML utuk mendefinisikan rangka kerja pemesanan terekstrensi di mana menyediakan konstruksi pesan yang dapat dipertukarkan pada protokol berbeda. Rangka kerja dirancang bebas dari model pemrograman dan spesifikasi implementasi semantik.
SOAP menjadi sangat mudah diterima oleh berbagai pihak – terutama oleh berbagai vendor TI – dikarenakan protokol ini memanfaatkan berbagai teknologi yang sudah ada sebelumnya dan sudah banyak digunakan. Misalnya untuk protokol transport, yang paling banyak digunakan adalah HTTP, walaupun dimungkinkan untuk menggunakan protokol transport lainnya. Sedangkan untuk format data atau message digunakan XML yang tidak diragukan lagi manfaat dan perannya di dalam pertukaran data. Dengan demikian, tidaklah terlalu mengherankan bila kemudian SOAP dianggap sebagai solusi penyelamat untuk mengatasi berbagai masalah yang dihadapi oleh teknologi – teknologi pendahulunya.

Implementasi SOAP sendiri kemudian disadari tidaklah mudah dan sesederhana yang dibayangkan. Karena SOAP sendiri tidak lebih dari
sekedar spesifikasi, yang mencoba untuk mendeskripsikan dan mendefinisikan setiap aspek dan mekanisme yang ada dalam sebuah RPC. Hal ini yang kemudian banyak membuat kerancuan dan kebingungan ketika seorang developer pertama kali mempelajari SOAP. Selain harus membiasakan diri dan menjadi fasih dengan spesifikasi SOAP, seorang developer juga harus fasih dengan spesifikasi dari tool atau program yang digunakan untuk menghasilkan sebuah message SOAP dan juga untuk membaca serta mengartikan sebuah message SOAP yang diterima. Benar, seorang developer akan memerlukan software atau tool untuk dapat berinteraksi dengan SOAP.

Di sinilah terjadi persaingan dalam hal implementasi SOAP, di mana beberapa vendor mengeluarkan seperangkat tool untuk menghasilkan dan menerima message SOAP seperti Microsoft yang mengeluarkan SOAP Toolkit dan sekarang sudah mencapai versi 2.0, Apache SOAP, PERL/SOAP Lite Client, dan sebagainya. Di sini terletak beberapa perbedaan dalam hal implementasi SOAP, di mana message SOAP yang dihasilkan oleh suatu toolkit belum tentu dapat diterima dan digunakan oleh pihak lain yang menggunakan platform berbeda. Banyak hal yang menyebabkan terjadinya masalah – masalah tersebut, dan tidak selalu hal tersebut merupakan masalah yang ada di SOAP itu sendiri, melainkan juga masalah – masalah dalam menginterpretasikan kode – kode syntax HTTP maupun syntax dari XML itu sendiri.

Komunikasi Client-server dengan pesan SOAP

1. Aplikasi klien mengirimkan pesan meminta SOAP untuk aplikasi server disepanjang jaringan
2. Berdasarkan URL yang diminta , server mengidentifikasi web service dalam keadaan invoke
3. Web service membaca pesan yg diminta SOAP dan mengidentifikasi operasi yang ingin dijalankan. Operasi menyesuaikan ke method dari penanggung komponen, untuk di invoke ke langkah selanjutnya. Konversi form XML ke java terjadi saat parameter permintaan dari pesan SOAP di web service untuk invoked operasi.
4. Web Service membaca pesan permintaan SOAP dan mengidentifikasi operasi yang dibutuhkan. Operasi ini akan melakukan korepodensi terhadap suatu metode dari komponen terakhir (back-end), yang kemudian akan panggil kemudian. Konversi dari XML ke java akan terjadi sebagai parameter permintaan atas pesan SOAP pada lapisan web service dalam operasi pemanggilan.
5. Metode komponen back-end yang cocok akan diminta oleh parameter Java yang lalu akan dipanggil kembali.
6. menjelang selesainya komponen back-end akan dikirim kembali sebagai respon web service serta kemudian akan dikonversi kembali menjadi bentuk XML dari bentuk java. Pemaketan tersebut dijadikan sebagai respon pesan SOAP.
7. Web Service akan mengirim kembali respon pesan SOAP ke aplikasi Client yang melakukan pemanggilan web service.

WSDL (Web Service Definition Language)
Format XML untuk mendeskripsikan antarmuka service beserta keterkaitannya dengan protokol tertentu. Biasanya digunakan untuk menggenerasi kode server atau klien atau untuk konfigurasi. WSDL mendeskripsikan web service yang aktif ke klien. Dengan menggunakan web service, aplikasi lain dapat mengakses sumber daya melalui jaringan dengan tren yang standar.

UDDI (Universal Description, Discovery, and Integration)
Protokol untuk mempublikasikan atau mencari metadata tentang web service sehingga aplikasi dapat menemukan web service.

F. APLIKASI ENTERPRISE

Enterprise menurut arti katanya mengacu pada organisasi atau individu atau entitas yang mempunyai kemungkinan bekerja sama untuk mencapai tujuan yang umum. Aplikasi enterprise adalah aplikasi besar yang memiliki banyak fungsi. Aplikasi enterprise merupakan aplikasi yang kompleks dengan berbagai syarat yang harus dipenuhinya agar aplikasi dapat berjalan dengan baik.

G. ISTILAH-ISTILAH SISTEM TERDISTRIBUSI

- **COM**

COM, Component Object Model adalah arsitektur aplikasi lunak yang memungkinkan terbentuknya aplikasi dari komponen aplikasi lunak binari. COM adalah arsitektur utama yang menjadi dasar bagi pelayanan aplikasi level tinggi, seperti pelayanan oleh OLE. Dengan demikian COM adalah kerangka dasar Microosoft dalam mengembangkan dan mendukung program obyek-komponen. Di dalam pemograman berorientasi obyek dan teknology obyek terdistribusi, sebuah komponen adalah program yang tidak digunakan dalam mengembangkan bagian yang dapat dikombinasikan dengan komponen lain yang ada pada komputer yang ada pada jaringan terdistribusi untuk membentuk aplikasi.

- **ActiveX**

ActiveX adalah nama yang diberikan Microsoft untuk strategi yang ada pada tool dan teknologi pemograman berorientasi obyek. Teknologi utamanya adalah Componen Object Model (COM), digunakan dalam jaringan dengan dukungan dari direktory, COM menjadi DCOM-Distributed Componen Object Model. Hal utama dalam membuat program, menkondisian lingkungan AktiveX adalah merupakan suatu komponen yang mampu menyesuaikan untuk bisa berjalan dengan baik. Komponen ini disebut ActiveX Control. ActiveX adalah tandingan dari Microsoft terhadap teknologi Java buatan Sun Microsystem.

- **DCOM**

DCOM (Distributed Componen Object Model) adalah kumpulan konsep Microsoft dan program interface dimana obyek program dari klien dapat meminta pelayanan dari server atas program obyek pada komputer di dalam jaringan. DCOM ini berbasis COM, dimana menyediakan sekumpulan interface yang memungkinkan klien dan server untuk melakukan komunikasi pada komputer yang sama.

- **COM+**

COM+ adalah pengembangan dari COM, sebagai suatu strateginya Microsoft dalam mengembangkan program tandingan dalam program aplikasi. Keduanya merupakan arsitektur pemograman beroentasi obyek serta sekumpulan pelayanan sistem operasi. COM+ memberikan tambahan pada COM suatu sistem pelayanan yang baru untuk aplikasi komponen ketika sedang berjalan. COM+ cenderung memberikan model guna memudahkan dalam membuat aplikasi bisnis pada Microsoft Transaction Server (MTS) pada Windows NT. Ini merupakan jawaban atas produk Sun Microsystem-IBM-Oracle yang dikenal dengan Enterprise Java Beans (EJB).