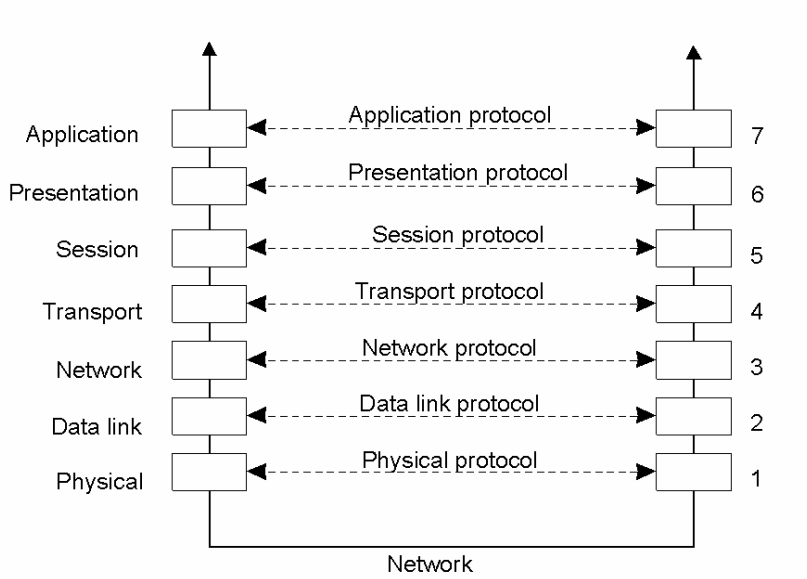
Bab 2. Komunikasi

I. Protokol

Protokol adalah sebuah aturan atau standar yang mengatur atau mengijinkan terjadinya hubungan, komunikasi, dan perpindahan data antara dua atau lebih titik komputer. Protokol dapat diterapkan pada perangkat keras, perangkat lunak atau kombinasi dari keduanya. Pada tingkatan yang terendah, protokol mendefinisikan koneksi perangkat keras.

Dahulu, komunikasi antar komputer dari vendor yang berbeda adalah sangat sulit dilakukan, karena mereka mengunakan protokol dan format data yang berbeda-beda. Sehingga International Standards Organization (ISO) membuat suatu arsitektur komunikasi yang dikenal sebagai Open System Interconnection (OSI), model yang mendefinisikan standar untuk menghubungkan komputer-komputer dari vendor-vendor yang berbeda. Model-OSI tersebut terbagi atas 7 layer.



Layer-layer tersebut disusun sedemikian sehingga perubahan pada satu layer tidak membutuhkan perubahan pada layer lain. Layer teratas (5, 6 and 7) adalah lebih cerdas dibandingkan dengan layer yang lebih rendah; Layer Application dapat menangani protocol dan format data yang sama yang digunakan oleh layer lain, dan seterusnya. Jadi terdapat perbedaan yang besar antara layer Physical dan layer Aplication.

Suatu permintaan, dihasilkan dari atas (contohnya Application Layer) diteruskan ke-enam layer di bawahnya yang setiap layer memiliki tugasnya masing-masing.

1. Physical Layer

Ini adalah layer yang paling sederhana berkaitan dengan electrical (dan optical) koneksi antar peralatan. Data biner dikodekan dalam bentuk yang dapat ditransmisi melalui media jaringan, sebagai contoh kabel, transceiver dan konektor yang berkaitan dengan layer Physical. Peralatan seperti repeater, hub dan network card adalah berada pada layer ini.

2. Data-link Layer

Layer ini sedikit lebih "cerdas" dibandingkan dengan layer physical, karena menyediakan transfer data yang lebih nyata. Sebagai penghubung antara media network dan layer protocol yang lebih high-level, layer data link bertanggung-jawab pada paket akhir dari data binari yang berasal dari level yang lebih tinggi ke paket diskrit sebelum ke layer physical. Akan mengirimkan frame (blok dari data) melalui suatu network.. Ethernet (802.2 & 802.3), Tokenbus (802.4) dan Tokenring (802.5) adalah protocol pada layer Data-link.

3. Network Layer

Tugas utama dari layer network adalah menyediakan fungsi routing sehingga paket dapat dikirim keluar dari segment network lokal ke suatu tujuan yang berada pada suatu network lain. IP, Internet Protocol, umumnya digunakan untuk tugas ini. Protocol lainnya seperti IPX, Internet Packet eXchange. Perusahaan Novell telah memprogram protokol menjadi beberapa, seperti SPX (Sequence Packet Exchange) & NCP (Netware Core Protocol). Protokol ini telah dimasukkan ke sistem operasi Netware. Beberaoa fungsi yang mungkin dilakukan oleh Layer Network ;

- Membagi aliran data biner ke paket diskrit dengan panjang tertentu
- Mendeteksi Error
- Memperbaiki error dengan mengirim ulang paket yang rusak
- Mengendalikan aliran

4. Transport Layer

Layer transport data, menggunakan protocol seperti UDP, TCP dan/atau SPX (Sequence Packet eXchange, yang satu ini digunakan oleh NetWare, tetapi khusus untuk koneksi berorientasi IPX). Layer transport adalah pusat dari mode-OSI. Layer ini menyediakan transfer yang reliable dan transparan antara kedua titik akhir, layer ini juga menyediakan multiplexing, kendali aliran dan pemeriksaan error serta memperbaikinya.

5. Session Layer

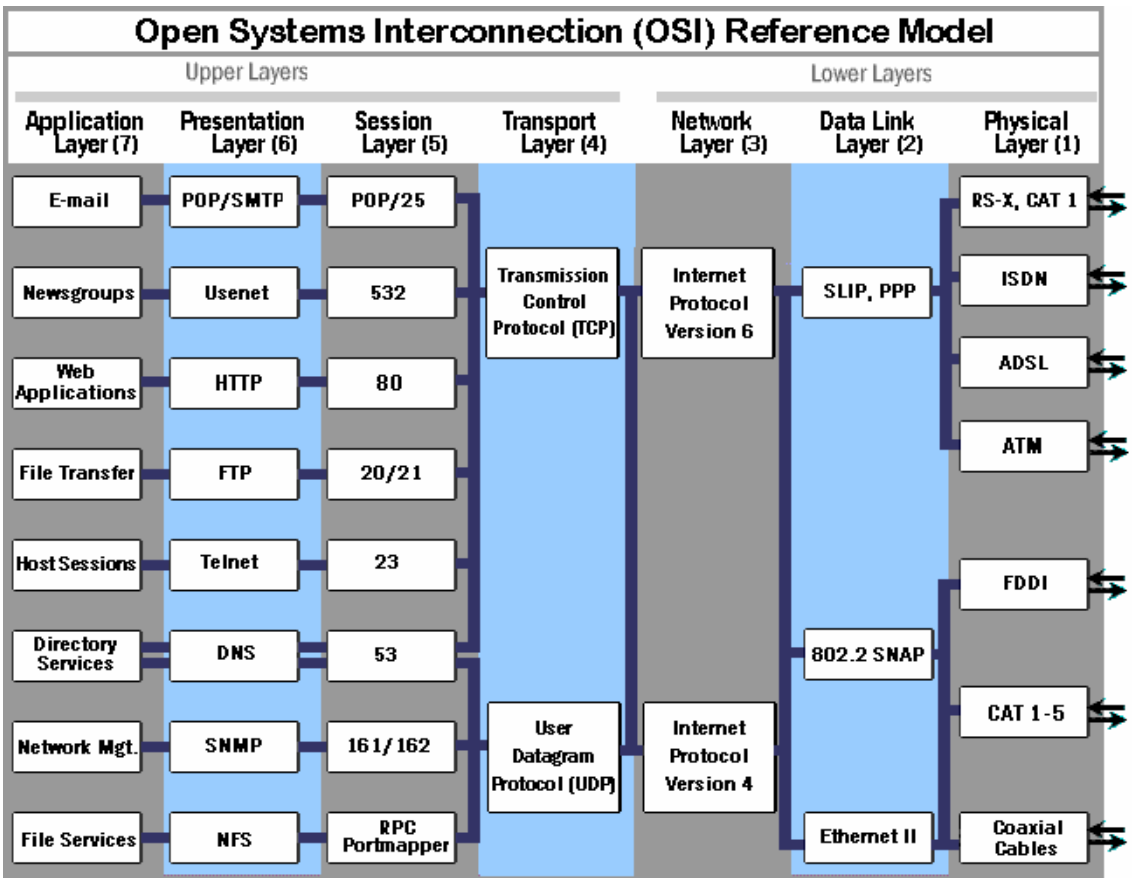
Layer Session, sesuai dengan namanya, sering disalah artikan sebagai prosedur logon pada network dan berkaitan dengan keamanan. Layer ini menyediakan layanan ke dua layer diatasnya, Melakukan koordinasi komunikasi antara entiti layer yang diwakilinya. Beberapa protocol pada layer ini: NETBIOS: suatu session interface dan protocol, dikembangkan oleh IBM, yang menyediakan layanan ke layer presentation dan layer application. NETBEUI, (NETBIOS Extended User Interface), suatu pengembangan dari NETBIOS yang digunakan pada produk Microsoft networking, seperti Windows NT dan LAN Manager. ADSP (AppleTalk Data Stream Protocol). PAP (Printer Access Protocol), yang terdapat pada printer Postscript untuk akses pada jaringan AppleTalk.

6. Presentation Layer

Layer presentation dari model OSI melakukan hanya suatu fungsi tunggal: translasi dari berbagai tipe pada syntax sistem. Sebagai contoh, suatu koneksi antara PC dan mainframe membutuhkan konversi dari EBCDIC character-encoding format ke ASCII dan banyak faktor yang perlu dipertimbangkan. Kompresi data (dan enkripsi yang mungkin) ditangani oleh layer ini.

7. Application Layer

Layer ini adalah yang paling 'cerdas', gateway berada pada layer ini. Gateway melakukan pekerjaan yang sama seperti sebuah router, tetapi ada perbedaan diantara mereka. Layer Application adalah penghubung utama antara aplikasi yang berjalan pada satu komputer dan resources network yang membutuhkan akses padanya.



II. Remote Procedure Call (RPC)

Remote Procedure Call (RPC) adalah sebuah metoda yang memungkinkan kita untuk mengakses sebuah prosedur yang berada di komputer lain. Untuk dapat melakukan ini sebuah komputer (server) harus menyediakan layanan remote prosedur. Pendekatan yang dilakukan adalah, sebuah server membuka socket, menunggu client yang meminta prosedur yang disediakan oleh server.

RPC masih menggunakan cara primitive dalam pemrograman, yaitu menggunakan paradigma procedural programming. Hal itu membuat kita sulit ketika menyediakan banyak remote procedure.

RPC menggunakan soket untuk berkomunikasi dengan proses lainnya. Pada sistem seperti SUN, RPC secara default sudah terinstall kedalam sistemnya, biasanya RPC ini digunakan untuk administrasi sistem. Sehingga seorang administrator jaringan dapat mengakses sistemnya dan mengelola sistemnya dari mana saja, selama sistemnya terhubung ke jaringan.

Umumnya protokol RPC yang digunakan pada saat ini adalah DCOM (Distributed Component Object Model). Saat ini ada alternatif protokol baru, yakni SOAP (Simple Object Access Protocol), yang berdasarkan pada teknologi XML.

Implementasi RPC

Sun Microsystems' Open Network Computing (ONC) : RPC specification, XDR (eXternal Data Representation) standard, UDP atau TCP transport protocol.

Xerox Courier : RPC model, Data representation standard, XNS (Xerox Network Systems) SPP (Sequenced Packet Protocol) sbg transport protocol, Apollo's Network Computing Architecture (NCA), RPC protocol, NDR (Network Data Representation).

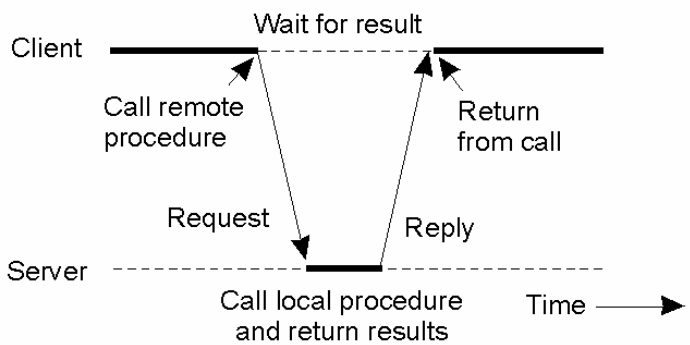
Kelebihan RPC

- Relatif mudah digunakan :
  Pemanggilan remote procedure tidak jauh berbeda dibandingkan pemanggilan local procedure. Sehingga pemrogram dapat berkonsentrasi pada software logic, tidak perlu memikirkan low level details seperti soket, marshalling & unmarshalling.
- Robust (Sempurna):
  Sejak th 1980-an RPC telah banyak digunakan dlm pengembangan mission-critical application yg memerlukan scalability, fault tolerance, & reliability.

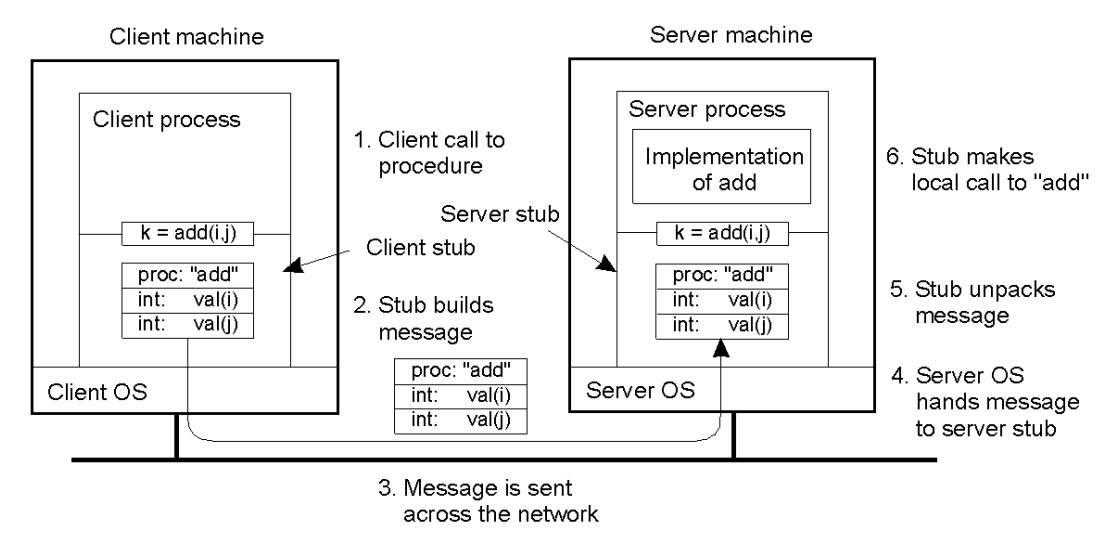
Kekurangan RPC

- Tidak fleksibel terhadap perubahan :
  - Static relationship between client & server at run-time.
  - Berdasarkan prosedural/structured programming yang sudah ketinggalan jaman dibandingkan OOP.
- Kurangnya location transparency :
  - Misalnya premrogram hanya boleh melakukan pass by value, bukan pass by reference.
  - Komunikasi hanya antara 1 klien & 1 server (one-to-one at a time).
  - Komunikasi antara 1 klien & beberapa server memerlukan beberapa koneksi yg terpisah.

Prinsip RPC dalam program clien-server



Langkah-langkah dalam RPC



1. Client procedure calls client stub in normal way
2. Client stub builds message, calls local OS
3. Client's OS sends message to remote OS
4. Remote OS gives message to server stub
5. Server stub unpacks parameters, calls server
6. Server does work, returns result to the stub
7. Server stub packs it in message, calls local OS
8. Server's OS sends message to client's OS
9. Client's OS gives message to client stub
10. Stub unpacks result, returns to client

Contoh Implementasi RPC dengan SUN RPC

- Fungsi yg dipanggil client melalui RPC:
  - bin_date_1:
    No arguments.
    Result in long integer: waktu saat ini, dlm jumlah detik sejak 00:00:00 GMT, January 1, 1970.
  - str_date_1
    Mengkonversi hasil bin_date_1 menjadi readable ASCII sring.
- Berkas yg dibuat programmer:
  - Server procedure: date_proc.c
  - RPC specification file: date.x
  - Client main function: rdate.c
- date.x -> rpcgen ->
  - Server stub: date_svc.c
  - date.h
  - Client stub: date_clnt.c
- Generate executable client program:
  cc –o rdate rdate.c date_clnt.c –lrpclib
- Generate executable server program:
  cc –o date_svc date_proc.c date_svc.c –lrpclib

III. Object Remote

Pendekatan kedua yang akan kita bahas adalah *Remote Method Invocation* (RMI), sebuah teknik pemanggilan method remote yang lebih secara umum lebih baik daripada RPC. RMI menggunakan paradigma pemrograman berorientasi obyek (OOP). Dengan RMI memungkinkan kita untuk mengirim obyek sebagai parameter dari *remote method*. Dengan dibolehkannya program java memanggil method pada remote obyek, RMI membuat pengguna dapat mengembangkan aplikasi java yang terdistribusi pada jaringan

Untuk membuat remote method dapat diakses RMI mengimplementasikan *remote object* menggukan stub dan skleton. Stub bertindak sebagai proxy disisi client, yaitu yang menghubungkan client dengan skleton yang berada disisi server. Stub yang ada disisi client bertanggung-jawab untuk membungkus nama method yang akan diakses, dan parameternya, hal ini biasa dikenal dengan marshalling. Stub mengirim paket yang sudah dibungkus ini ke server dan akan di buka (*unmarshalling*) oleh skleton. Skleton akan menerima hasil keluaran yang telah diproses oleh method yang dituju, lalu akan kembali dibungkus (marshall) dan dikirim kembali ke client yang akan diterima oleh stub dan kembali dibuka paketnya (unmarshall).

Untuk membuat remote obyek kita harus mendefinisikan semua method yang akan kita sediakan pada jaringan, setelah itu dapat digunakan RMI compiler untuk membuat stub dan skleton. Setelah itu kita harus mem-binding remote obyek yang kita sediakan kedalam sebuah RMI registry. Setelah itu client dapat mengakses semua remote method yang telah kita sediakan menggunkan stub yang telah dicompile menggunakan RMI compiler terebut.

Akses ke Obyek *Remote*

Sekali obyek didaftarkan ke server, client dapat mengakses remote object dengan menjalankan *Naming.lookup()* method. RMI menyediakan url untuk pengaksesan ke remote obyek yaitu *rmi://host/obyek*, dimana host adalah nama server tempat kita mendaftarkan remote obyek dan obyek adalah parameter yang kita gunakan ketika kita memanggil method *Naming.rebind()*. Client juga harus menginstall RMISecurityManager untuk memastikan keamanan client ketika membuka soket ke jaringan.

Java memiliki sistem security yang baik sehingga user dapat lebih nyaman dalam melakukan komunikasi pada jaringan. Selain itu java sudah mendukung pemorograman berorientasi object, sehingga pengembangan software berskala besar sangat dimungkinkan dilakukan oleh java. RMI sendiri merupakan sistem terdistribusi yang dirancang oleh SUN pada platfrom yang spesifik yaitu Java, apabila anda tertarik untuk mengembangkan sistem terdistribusi yang lebih portable dapat digunakan CORBA sebagai solusi alternatifnya.