

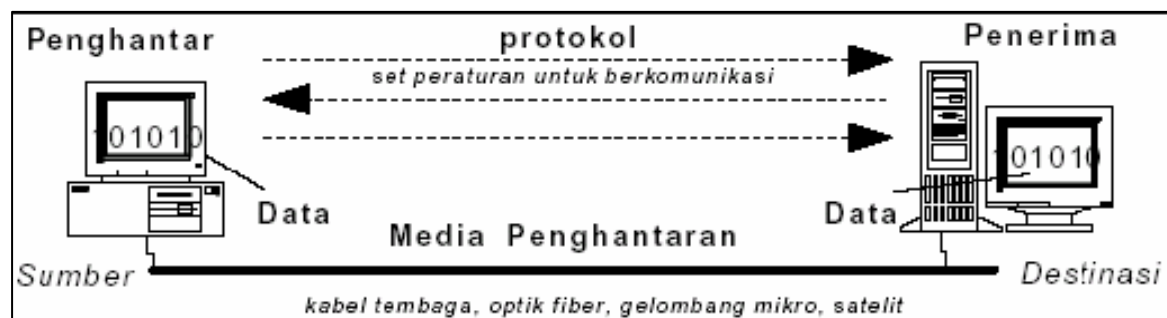
Bab 2. Komunikasi

2.1. Komunikasi Data

Komunikasi data adalah merupakan bagian dari telekomunikasi yang secara khusus berkenaan dengan transmisi atau pemindahan data dan informasi diantara komputer-komputer dan piranti-piranti yang lain dalam bentuk digital yang dikirimkan melalui media komunikasi data. Data berarti informasi yang disajikan oleh isyarat digital. Komunikasi data merupakan baguan vital dari suatu masyarakat informasi karena sistem ini menyediakan infrastruktur yang memungkinkan komputer-komputer dapat berkomunikasi satu sama lain.

Komponen Komunikasi Data

- **Pengirim**, adalah piranti yang mengirimkan data
- **Penerima**, adalah piranti yang menerima data
- **Data**, adalah informasi yang akan dipindahkan
- **Media pengiriman**, adalah media atau saluran yang digunakan untuk mengirimkan data
- **Protokol**, adalah aturan-aturan yang berfungsi untuk menyelaraskan hubungan.



2.2. Protokol

Protokol dapat diartikan sebagai sebuah aturan yang mendefinisikan beberapa fungsi yang ada dalam sebuah jaringan komputer, misalnya mengirim pesan, data, informasi dan fungsi lain yang harus dipenuhi oleh sisi pengirim dan sisi penerima agar komunikasi dapat berlangsung dengan benar, walaupun sistem yang ada dalam jaringan tersebut berbeda sama sekali. Protokol ini mengurus perbedaan format data pada kedua sistem hingga pada masalah koneksi listrik.

Komponen Protokol

1. Aturan atau prosedur, mengatur pembentukan/pemutusan hubungan
2. Format atau bentuk, mengatur proses transfer data representasi pesan
3. Kosakata (vocabulary), jenis pesan dan makna masing-masing pesan

Fungsi Protokol

Secara umum fungsi dari protokol adalah untuk menghubungkan sisi pengirim dan sisi penerima dalam berkomunikasi serta dalam bertukar informasi agar dapat berjalan dengan baik dan benar. Sedangkan fungsi protokol secara detail dapat dijelaskan berikut:

- *Fragmentasi dan reassembly*
Fungsi dari fragmentasi dan *reassembly* adalah membagi informasi yang dikirim menjadi beberapa paket data pada saat sisi pengirim mengirimkan informasi dan setelah diterima maka sisi penerima akan menggabungkan lagi menjadi paket informasi yang lengkap.
- *Encapsulation*
Fungsi dari *encapsulation* adalah melengkapi informasi yang dikirimkan dengan *address*, kode-kode koreksi dan lain-lain.
- *Connection control*
Fungsi dari *connection control* adalah membangun hubungan (*connection*) komunikasi dari sisi pengirim dan sisi penerima, dimana dalam membangun hubungan ini juga termasuk dalam hal pengiriman data dan mengakhiri hubungan.
- *Flow control*
Befungsi sebagai pengatur perjalanan data dari sisi pengirim ke sisi penerima.
- *Error control*
Dalam pengiriman data tak lepas dari kesalahan, baik itu dalam proses pengiriman maupun pada waktu data itu diterima. Fungsi dari *error control* adalah mengontrol terjadinya kesalahan yang terjadi pada waktu data dikirimkan.
- *Transmission service*
Fungsi dari *transmission service* adalah memberi pelayanan komunikasi data khususnya yang berkaitan dengan prioritas dan keamanan serta perlindungan data.

Susunan Protokol

Protokol jaringan disusun oleh dalam bentuk lapisan-lapisan (*layer*). Hal ini mengandung arti supaya jaringan yang dibuat nantinya tidak menjadi rumit. Di dalam layer ini, jumlah, nama, isi dan fungsi setiap layer berbeda-beda. Akan tetapi tujuan dari setiap layer ini adalah memberi layanan ke layer yang ada di atasnya.

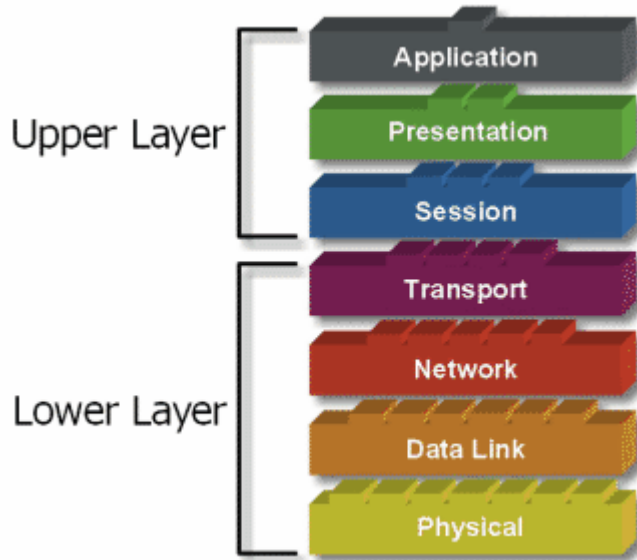
Susunan dari layer menunjukkan tahapan dalam melakukan komunikasi. Antara setiap layer yang berdekatan terdapat sebuah interface. Interface menentukan layanan layer yang di bawah kepada layer yang di atasnya.

Pada saat merencanakan sebuah jaringan, hendaknya memperhatikan bagaimana menentukan interface yang tepat yang akan ditempatkan di antara dua layer yang bersangkutan.

Standarisasi Protokol (ISO 7498)



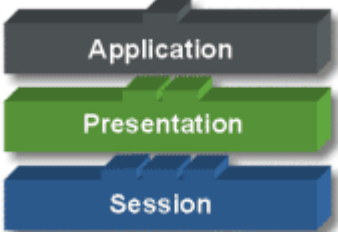
Dahulu, komunikasi antar komputer dari vendor yang berbeda adalah sangat sulit dilakukan, karena mereka menggunakan protokol dan format data yang berbeda-beda. Sehingga *International Standards Organization* (ISO) membuat suatu arsitektur komunikasi yang dikenal sebagai *Open System Interconnection*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




Model Layer OSI dibagi dalam dua group: “upper layer” dan “lower layer”. “Upper layer” fokus pada aplikasi pengguna dan bagaimana file direpresentasikan di komputer. “Lower layer” adalah intisari komunikasi data melalui jaringan aktual.



Gambar 1: Pembagian Model Layer OSI

Penjelasan Model OSI Layer:

Model OSI	Keterangan
	<p>Application Layer: Menyediakan jasa untuk aplikasi pengguna. Layer ini bertanggungjawab atas pertukaran informasi antara program komputer, seperti program e-mail, dan service lain yang jalan di jaringan, seperti server printer atau aplikasi komputer lainnya.</p>
	<p>Presentation Layer: Bertanggung jawab bagaimana data dikonversi dan diformat untuk transfer data. Contoh konversi format text ASCII untuk dokumen, .gif dan JPG untuk gambar. Layer ini membentuk kode konversi, translasi data, enkripsi dan konversi.</p>
	<p>Session Layer: Menentukan bagaimana dua terminal menjaga, memelihara dan mengatur koneksi,- bagaimana mereka saling berhubungan satu sama lain. Koneksi di layer ini disebut "session".</p>

	<p>Transport Layer: Bertanggung jawab membagi data menjadi segmen, menjaga koneksi logika “end-to-end” antar terminal, dan menyediakan penanganan error (error handling). <i>Transport layer</i> berfungsi untuk menerima data dari <i>session</i> layer, memecah data menjadi bagian-bagian yang lebih kecil, meneruskan data ke <i>network layer</i> dan menjamin semua potongan data tersebut bisa tiba di sisi penerima dengan benar.</p>
	<p>Network Layer: Bertanggung jawab menentukan alamat jaringan, menentukan rute yang harus diambil selama perjalanan, dan menjaga antrian trafik di jaringan. Data pada layer ini berbentuk paket.</p> <ul style="list-style-type: none"> • Fungsi utama dari layer network adalah pengalamatan dan routing. Pengalamatan pada layer network merupakan pengalamatan secara logical • Routing digunakan untuk pengarah jalur paket data yang akan dikirim. • Transport dari suatu informasi
	<p>Data Link Layer: Menyediakan link untuk data, memaketkannya menjadi frame yang berhubungan dengan “hardware” kemudian diangkut melalui media. komunikasinya dengan kartu jaringan, mengatur komunikasi layer physical antara sistem koneksi dan penanganan error. Tugas utama data link layer adalah sebagai fasilitas transmisi raw data dan mentransformasi data tersebut ke saluran yang bebas dari kesalahan transmisi.</p> <p>Fungsi yang diberikan pada layer data link antara lain :</p> <ul style="list-style-type: none"> • Arbitration, pemilihan media fisik untuk penentuan waktu pengiriman data, metode yang dipakai CSMA/CD(<i>Carrier Sense Multiple Access /Collision Detection</i>). • Addressing, pengalamatan bersifat fisik yaitu dgn MAC(media Access Control) yang ditanamkan pada interface perangkat jaringan. • Error detection, menentukan apakah data telah berhasil terkirim, tekniknya FCS(Frame Check Sequence) dan CRC(Cyclic Redundancy Check) • Identify Data Encapsulation.



Physical Layer: Bertanggung jawab atas proses data menjadi bit dan mentransfernya melalui media, seperti kabel, dan menjaga koneksi fisik antar sistem. Layer ini mengatur tentang bentuk interface yang berbeda-beda dari sebuah media transmisi. Spesifikasi yang berbeda misal konektor, pin, penggunaan pin, arus listrik yang lewat, encoding, sumber cahaya dll. Secara umum masalah-masalah desain yang ditemukan di sini berhubungan secara mekanik, elektrik dan interface prosedural, dan media fisik yang berada di bawah physical layer. Contoh : EIA/TIA-232, V35, EIA/TIA- 449, V.24, RJ45, Ethernet, NRZI, NRZ, B8ZS

Hubungan antara model referensi OSI dengan protokol internet dapat dilihat dalam tabel di bawah ini.

Model OSI		TCP/IP	Protokol TCP/IP	
No	Lapisan		Nama Protokol	Kegunaan
7	Aplikasi	Aplikasi	DHCP (<i>Dynamic Host Configuration Protocol</i>)	Protokol untuk distribusi IP pada jaringan untuk jumlah IP terbatas
			DNS (<i>Domain Name Server</i>)	Database nama domain mesin dan nomer IP
			FTP (<i>File Transfer Protocol</i>)	Protokol untuk transfer file
			HTTP (<i>HyperText Transfer Protocol</i>)	Protokol untuk transfer file HTML dan web
			MIME ((<i>Multipurpose Internet Mail Extention</i>)	Protokol untuk mengirim file binary dalam bentuk teks
			NNTP (<i>Network News Transfer Protocol</i>)	Protokol untuk menerima dan mengirim newsgroup
			POP (<i>Post Office Protocol</i>)	Protokol untuk mengambil mail dari server
6	Presentasi		SMB (<i>Server Message Block</i>)	Protokol untuk transfer berbagai server file DOS dan Windows
			SMTP (<i>Simple Mail Transfer Protocol</i>)	Protokol untuk pertukaran mail
			SNMP (<i>Simple Network Management Protocol</i>)	Protokol untuk manajemen jaringan
			Telnet	Protokol untuk akses dari jarak jauh
5	Sessi		TFTP (<i>Trivial FTP</i>)	Protokol untuk transfer file
			NETBIOS (<i>Network Basic Input Output System</i>)	BIOS jaringan standar
			RPC (<i>Remote Procedure Call</i>)	Prosedur pemanggilan jarak jauh
		Socket	Input Output untuk network jenis BSD-UNIX	

4	Transport	Transport	TCP (<i>Transmission Control Protocol</i>)	Protokol pertukaran data berorientasi (<i>connection oriented</i>)
			UDP (<i>User Datagram Protocol</i>)	Protokol pertukaran data non-orientasi (<i>connectionless</i>)
3	Network	Internet	IP (<i>Internet Protocol</i>)	Protokol untuk menetapkan routing
			RIP (<i>Routing Information Protocol</i>)	Protokol untuk memilih routing
			ARP (<i>Address Resolution Protocol</i>)	Protokol untuk mendapatkan informasi hardware dari nomer IP
			RARP (<i>Reverse ARP</i>)	Protokol untuk mendapatkan informasi nomer IP dari hardware
2	Datalink	LLC (<i>Logical Link Control</i>)	PPP (<i>Point to Point Protocol</i>)	Protokol untuk poin ke poin
		MAC (<i>Media Access Control</i>)		
1	Fisik	Network Interface	Ethernet, FDDI, ISDN, ATM	

2.3. Remote Procedure Call (RPC)

Remote Procedure Call (RPC) adalah sebuah metode yang memungkinkan kita untuk mengakses sebuah prosedur yang berada di komputer lain. Untuk dapat melakukan ini sebuah *server* harus menyediakan layanan *remote procedure*. Pendekatan yang dilakukan adalah sebuah *server* membuka *socket*, lalu menunggu *client* yang meminta prosedur yang disediakan oleh server. Bila *client* tidak tahu harus menghubungi *port* yang mana, *client* bisa *me-request* kepada sebuah *matchmaker* pada sebuah *RPC port* yang tetap. *Matchmaker* akan memberikan *port* apa yang digunakan oleh prosedur yang diminta *client*.

RPC masih menggunakan cara primitif dalam pemrograman, yaitu menggunakan paradigma *procedural programming*. Hal itu membuat kita sulit ketika menyediakan banyak *remote procedure*. RPC menggunakan *socket* untuk berkomunikasi dengan proses lainnya. Pada sistem seperti SUN, RPC secara *default* sudah ter-*install* 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.

Kelebihan RPC

- Relatif mudah digunakan :
Pemanggilan *remote procedure* tidak jauh berbeda dibandingkan pemanggilan *local procedure*. Sehingga pemrogram dapat berkonsentrasi pada *software logic*,

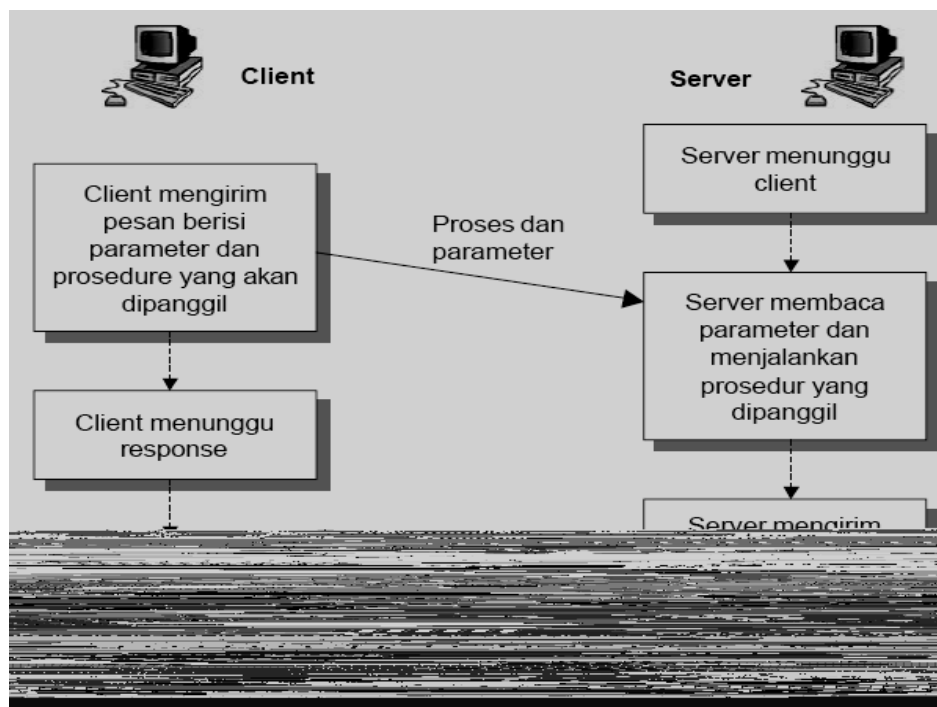
tidak perlu memikirkan low level details seperti *socket*, *marshalling* & *unmarshalling*.

- Robust (Sempurna):
Sejak th 1980-an RPC telah banyak digunakan dlm pengembangan mission-critical application yg memerlukan *scalability*, *fault tolerance*, & *reliability*.

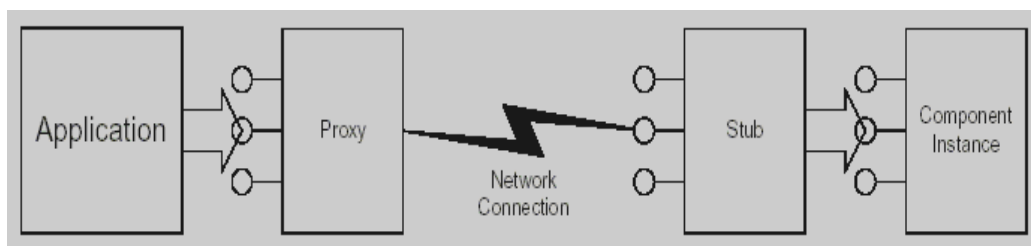
Kekurangan RPC

- Tidak fleksibel terhadap perubahan:
 - *Static relationship between client & server at run-time.*
- Berdasarkan prosedural/structured programming yang sudah ketinggalan jaman dibandingkan OOP.

Prinsip RPC dalam program Client-Server



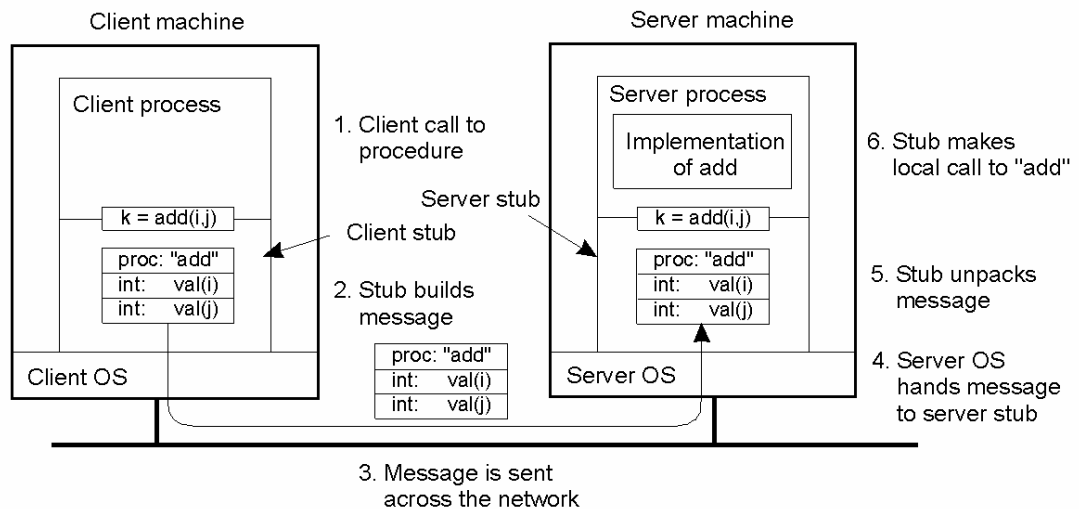
Skema RPC ini dilakukan juga pada proses-proses yang running di komputer berlainan



- Sebelum mekanisme RPC digunakan, data harus di-packaging ke dalam format transmisi. Langkah ini dinamakan *marshalling*
- *Proxy* bertanggung jawab untuk *marshalling* data, kemudian mengirimkan data dan meminta instans dari komponen (*remote*)

- *Stub* menerima request, unmarshall data, dan memanggil method yang diminta. Kemudian proses mengembalikan nilai yang diinginkan

Langkah-langkah dalam RPC



1. Prosedur client memanggil client stub
2. Client stub membuat pesan dan memanggil OS client
3. OS client mengirim pesan ke OS server
4. OS server memberikan pesan ke server stub
5. Server stub meng-*unpack* parameter-parameter untuk memanggil server
6. Server mengerjakan operasi, dan mengembalikan hasilnya ke server stub
7. Server stub mem-*pack* hasil tsb dan memanggil OS server
8. OS server mengirim pesan (hasil) ke OS client
9. OS client memberikan pesan tersebut ke client stub
10. Client stub meng-*unpack* hasil dan mengembalikan hasil tersebut ke client

2.4. Object Remote

Meskipun teknologi RPC ini relatif sudah memberikan kenyamanan bagi developer, tapi perkembangan yang terjadi di bidang pemrograman berorientasi objek akhirnya menuntut kehadiran teknologi baru. Sederet teknologi akhirnya benar-benar muncul, antara lain; RMI (*Remote Method Invocation*), CORBA (*Common Object Request Broker Architecture*), dan SOAP (*Simple Object Access Protocol*).

Remote Method Invocation (RMI) adalah sebuah teknik pemanggilan method remote yang lebih secara umum lebih baik daripada RPC. RMI menggunakan paradigma pemrograman berorientasi obyek (*Object Oriented Programming*). 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.

Aplikasi RMI seringkali terdiri dari dua program terpisah yaitu server dan client. Aplikasi server semacam ini biasanya membuat beberapa objek remote, menyediakan

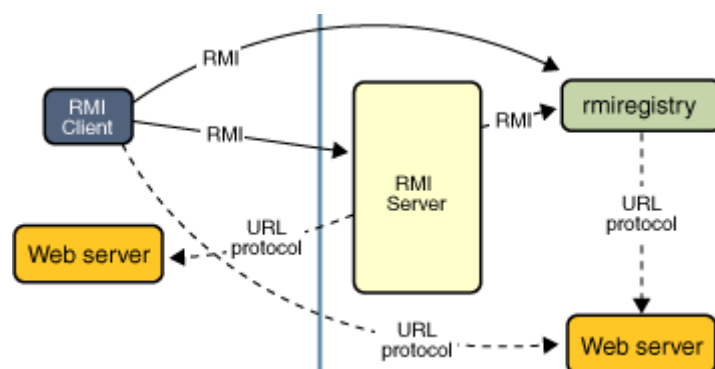
referensi terhadap objek-objek tersebut sehingga dapat diakses, serta menunggu client menginvoke/memanggil method dari objek-objek remote tersebut. Aplikasi client mendapatkan referensi remote ke satu atau lebih objek remote di server dan menjalankan method dari objek tersebut.

RMI menyediakan mekanisme dimana server dan client berkomunikasi dan memberikan informasi secara timbal balik. Aplikasi semacam ini seringkali disebut aplikasi objek terdistribusi.

Aplikasi objek terdistribusi seringkali melakukan hal berikut:

- Melokasikan objek remote: Aplikasi dapat menggunakan satu dari dua mekanisme untuk mendapatkan referensi ke objek remote. Aplikasi dapat mendaftarkan objek remote dengan fasilitas penamaan RMI (*naming facility*) yaitu *rmiregistry* atau aplikasi dapat mem-*pass* dan mengembalikan referensi objek remote sebagai bagian dari operasi normal.
- Berkomunikasi dengan objek remote: Detail dari komunikasi antara objek remote ditangani oleh RMI, bagi programmer komunikasi remote tampak seperti invokasi method Java standar.
- Memanggil (*load*) bytecode untuk objek yang di-*pass*: Karena RMI mengizinkan pemanggil (*caller*) untuk mem-*pass* objek ke objek remote, RMI menyediakan mekanisme yang diperlukan objek me-*load* kode objek, sebagaimana juga mentransmisikan datanya.

Ilustrasi berikut menggambarkan aplikasi RMI terdistribusi yang menggunakan registry untuk mendapatkan referensi ke objek remote. Server memanggil registry untuk mengasosiasikan (mengikat) suatu nama dengan objek remote. Client mencari objek remote dengan namanya pada registry server dan meng-*invoke* method dari objek. Ilustrasi ini juga menunjukkan sistem RMI menggunakan Web server untuk memanggil class bytecodes, dari server ke client dan dari client ke server, untuk objek-objek yang diperlukan.



Langkah-Langkah Pembuatan Program dengan RMI

Dalam *RMI*, semua informasi tentang satu pelayanan *server* disediakan dalam suatu definisi *remote interface*. Dengan melihat pada definisi *interface*, seorang pemrogram dapat memberitahukan *method* apa yang dapat dikerjakan oleh *server*, meliputi data apa yang diterima dan data apa yang akan dikirim sebagai tanggapan.

Definisi yang ada pada *remote interface* menentukan karakteristik *methods* yang disediakan *server* yang dapat dilihat oleh *client*. *Client programmer* harus dapat

mengetahui *methods* apa yang disediakan *server* dan bagaimana memanggilnya langsung dengan melihat ke *remote interface*. *Client* mendapatkan referensi ke *remote object* melalui *RMI registry*.

Membangun suatu aplikasi terdistribusi menggunakan *RMI* meliputi 6 langkah. Keenam langkah tersebut adalah:

1. Mendefinisikan *remote interface*
2. Implementasi *remote interface* dan *server*
3. Pengembangan *client* (atau *applet*) yang menggunakan *remote interface*
4. Mengkompilasi *source files* dan mem-buat *stub* and *skeletons*
5. Memulai (*start*) *RMI registry*
6. Menjalankan *server* dan *client*

Referensi:

1. Rijal Fadilah, Handout Komunikasi Data dan Jaringan Komputer, <http://rijalfadilah.multiply.com/>
2. Trindiana dkk., Tugas Kuliah Pengantar Sistem Terdistribusi, 2008.
3. Mudji, Model Jaringan 7 OSI Layer, <http://mudji.net/press/?p=61>
4. Lintang, Pengenalan Hardware dan Topologi Jaringan Komputer, <http://staffsite.gunadarma.ac.id/lintang/index.php?stateid=download&id=2268&part=files>
5. <http://bebas.vlsm.org/v06/Kuliah/SistemOperasi/BUKU/SistemOperasi-4.X-1/ch17s06.html>
6. www.bebas.vlsm.org/v06/Kuliah/SistemOperasi/BUKU/bahan/bahan-bab3.pdf
7. Isak Rickyanto, Tutorial Pengenalan Java RMI, <http://www.benpinter.net/article.php?story=20030818005713433>
8. Ayu Anggriani dkk., Tugas Kuliah Pengantar Sistem Terdistribusi, 2008.