Objek Terdistribusi dan Remote Invocation

I Made Andhika, S.Kom

Middleware

- Remote Procedure Call (RPC)
  - client memanggil sebuah prosedur dan menjalankan pada komputer lain
  - Pemanggilan tersebut sama seperti pemanggilan lokal
- Remote Method Invocation (RMI)
  - Objek lokal memanggil method dari objek yang berada di komputer lain
  - Pemanggilan tersebut sama seperti pemanggilan lokal
- Event-based Distributed Programming
  - Objek menerima “pemberitahuan” (notification) suatu event yang terjadi pada komputer/proses lain

Ciri-ciri Transparansi Middleware

- Transparansi Lokal
  - Pemanggilan pada RMI dan RPC tanpa mengetahui lokasi method/prosedur yang dipanggil
- Transparansi Protokol Transport
  - Protokol request/reply digunakan untuk penerapan RPC dapat menggunakan beberapa protokol transport
- Transparansi Platform
  - Menggunakan presentasi data eksternal
- Transparansi Bahasa Pemrograman
  - Dengan menggunakan bahasa yang tidak tergantung bahasa pemrograman, yaitu *Interface Definition Languages, seperti IDL* CORBA

Antarmuka RMI dan RPC

- Antarmuka (interface) didefinisikan untuk setiap objek
- Menyediakan definisi sekumpulan method
- Menyembunyikan semua rinci implementasi tiap method
- Pengaksesan hanya dapat terjadi melalui method yang terdefinisikan pada antarmuka
- Khusus untuk antarmuka pada sistem terdistribusi
  - Tidak ada akses langsung ke remote variabel
    - Menggunakan mekanisme *message passing untuk* pengiriman data objek dan variabel
      - Menerapkan *Protokol request-reply*
  - Mekanisme *passing paramater lokal (by value, by reference)* tidak dapat diterapkan untuk pemanggilan remote
    - Harus menggunakan format request dan reply
  - Pointer tidak valid untuk *remote address space*
    - Pointer adalah untuk *local address space*

Antarmuka RMI dan RPC

- Antarmuka RPC dan RMI sering ditinjau sebagai sistem client/server
  - service interface (dalam model client server)
    - spesifikasi prosedur dan method yang ditawarkan oleh server
  - remote interface (dalam model RMI)
    - Spesifikasi method sebuah objek yang dipanggil oleh objek dari proses lainnya

Interface Definition Language

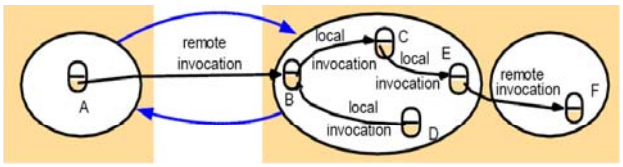
- Tidak mungkin mengakses secara langsung variabel dalam class remote
- Oleh karena itu, akses hanya melalui suatu antarmuka
- IDL dikompilasi untuk menyesuaikan dengan bahasa pemrograman yang digunakan
- Contoh: CORBA IDL

```
// In file Person.idl
struct Person {
        string name;
        string place;
        long year;
};
interface PersonList {
        readonly attribute string listname;
        void addPerson(in Person p) ;
        void getPerson(in string name, out Person p);
        long number();
};
```

Komponen Model Objek

- Referensi Objek
  - Menunjuk pada objek sebuah class
  - Dapat dinyatakan ke variabel, passing sebagai argumen dan dihasilkan dari method
- Antarmuka
  - Mendefinisikan method, parameter, tipe pengembalian method
- Aksi (method)
  - Dimulai dengan pemanggilan method dari objek lain
  - Pengaruh yang mungkin
- State objek yang dipanggil methodnya mungkin berubah
- Dapat mempengaruhi pemanggilan method berikutnya
- Exception
  - Reaksi dari sistem atau pemakai terhadap situasi kesalahan atau yang tidak diharapkan
- Garbage Collection
  - Pembuangan *resource (memori) yang digunakan objek yang* tidak direferensikan oleh objek dan tidak digunakan lagi

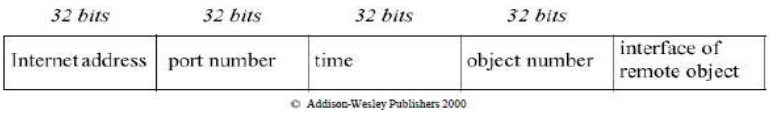
Model Objek Terdistribusi



- Mengadopsi model client/server
  - Server memelihara objek dan membolehkan remote objek untuk mengakses method objek tersebut yang dikhususkan untuk pemakaian definisi antarmuka
  - Client memanggil method yang ditawarkan oleh server menggunakan remote method invocation (RMI)
- RMI menerapkan protokol request/reply
  - Request membawa referensi remote method dan remote objek dalam suatu parameter ke server
  - reply mengembalikan parameter (hasil dari RMI) ke client

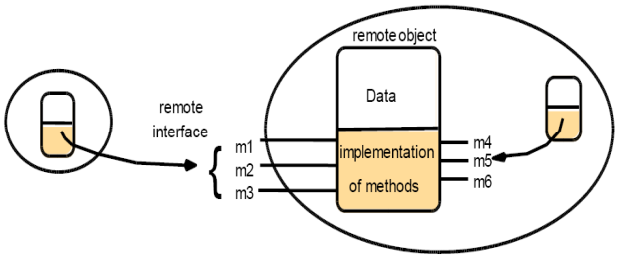
Model RMI – Object Reference

- Pada gambar slide sebelumnya, objek B dan F adalah objek remote yang menerima pemanggilan RMI
- Dalam setiap pemanggilan, client mendasarkan pada referensi objek remote (objek server) yang memiliki format umum :



- Dapat dipertukarkan sebagai parameter atau hasil suatu method

Model RMI – Object Reference

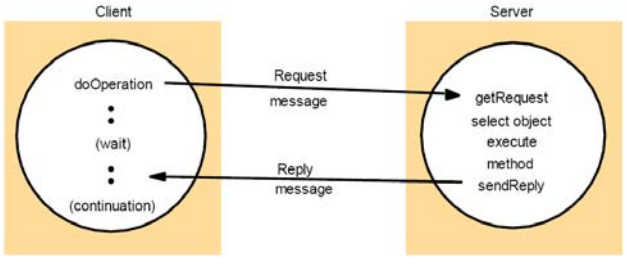


- Remote interface – mendefinisikan method apa saja yang dapat diakses oleh client
- Pada contoh gambar slide 8, objek B dan F harus memiliki remote interface
- Contoh :
  - CORBA menggunakan IDL untuk definisi Interface objek server
  - RMI menggunakan interface Java untuk definisi objek server

Model RMI – Else ?

- **Actions**
  - Disebabkan oleh adanya pemanggilan method
  - Jika pemanggilan terjadi antar proses atau komputer yang berbeda, maka RMI digunakan untuk menyebabkan adanya action.
  - Untuk menggunakan RMI, sekali lagi client harus butuh menyimpan referensi objek server. Contoh gambar slide 8, objek A harus menyimpan referensi objek server B
  - Contoh lain : objek A dapat memanggil method objek F melalui objek B
- **Garbage Collection**
  - Menerapkan *distributed garbage collection, yaitu koordinasi* antara garbage collection lokal dengan remote
- **Exception**
  - RMI harus mampu membangkitkan exception jika ada suatu kesalahan sewaktu invocation, seperti server crash, server terlalu sibuk (timeout), client crash
  - Mampu melakukan recovery terhadap kesalahan

Protokol Request/Reply untuk RMI



public byte[] doOperation (RemoteObjectRef o, int methodId, byte[] arguments)
sends a request message to the remote object and returns the reply. The arguments specify the remote object, the method to be invoked and the arguments of that method.

public byte[] getRequest ();
acquires a client request via the server port.

public void sendReply (byte[] reply, InetAddress clientHost, int clientPort);
sends the reply message reply to the client at its Internet address and port.

Masalah Perancangan untuk RMI

- **Semantik Pemanggilan RMI**
  - doOperation() dapat diterapkan dalam beberapa cara untuk menyediakan jaminan pengiriman yang berbeda
    - *Retry request message : pengiriman ulang request* walaupun reply diterima atau server diasumsikan gagal
    - *Duplicate filtering : apakah dilakukan filtering terhadap* request yang sama atau tidak
  - Ketika semua request yang dikirim ulang diterima server, ada dua kemungkinan yang dilakukan objek server
    - *Retransmission of result : apakah mampu menyimpan* (caching) hasil reply, sehingga jika memang perlu melakukan retransmisi ulang tidak perlu melakukan eksekusi kembali
    - *Repeated execution prosedur*
  - Dari dua kemungkinan-kemungkinan dua operasi tersebut, memunculkan beberapa semantik invocation

Semantik Pemanggilan RMI

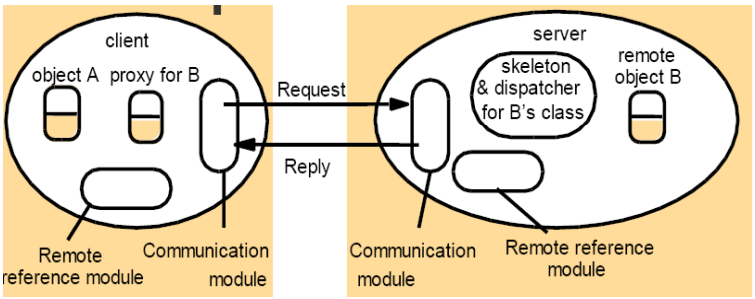
Fault tolerance measures			*Invocation semantics*
Retransmit request message	*Duplicate filtering*	*Re-execute procedure or retransmit reply*	
No	Not applicable	Not applicable	*Maybe*
Yes	No	Re-execute procedure	*At-least-once*
Yes	Yes	Retransmit reply	*At-most-once*

- **Maybe**
  - Pemanggil tidak mengetahui apapun apakah operasi dieksekusi atau tidak. Di sini tidak ada mekanisme fault tolerance

Semantik Pemanggilan RMI

- **At-least-once**
  - Pemanggil (invoker) menerima :
    - Hasil, atau
    - Exception, jika tidak ada hasil yang diterima
  - Jika ada hasil yang diterima, berarti pemanggil mengetahui bahwa operasi telah dilakukan paling tidak sekali, namun tidak mengetahui jika exception dihasilkan
- **At-most-once**
  - Pemanggil menerima :
    - Hasil, pemanggil mengetahui bahwa operasi telah dilakukan sekali, atau
    - Exception yang memberikan informasi bahwa tidak ada hasil yang diterima, dalam hal ini operasi dilakukan sekali atau tidak sama sekali
  - Contoh : JavaRMI dan CORBA
  - CORBA memungkinkan semantik maybe ketika tidak ada hasil yang dikembalikan

Implementasi RMI



- **Communication module**
  - request/reply message
  - Di server :
    - Memilih dispatcher untuk class objek yang dipanggil berdasar objek reference dari invoker
    - Menghasilkan lokal reference dari remote reference module

Implementasi RMI

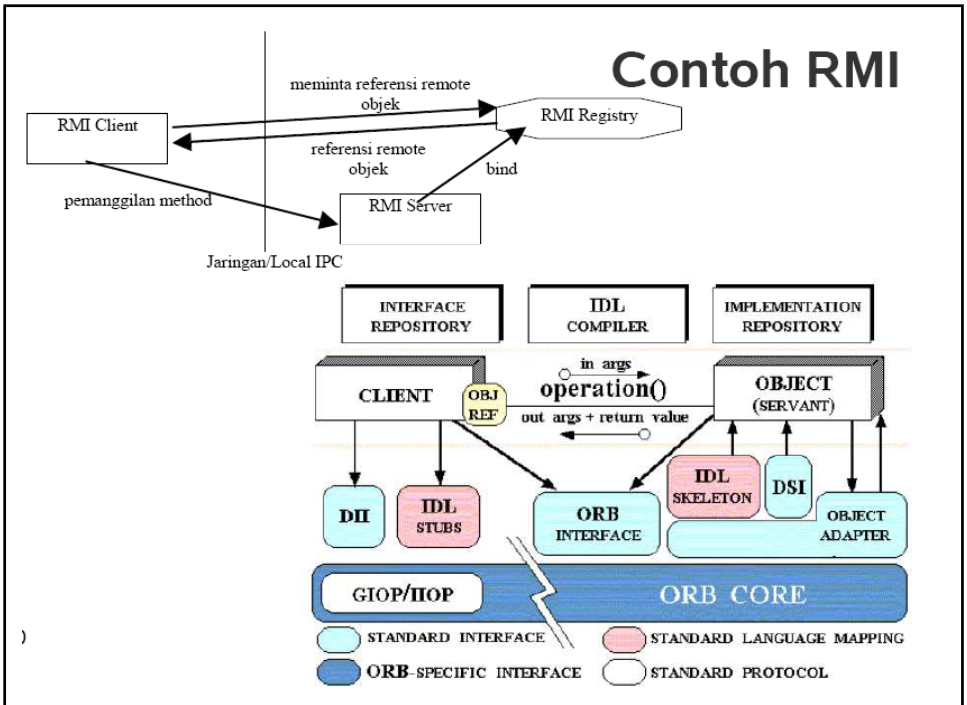
- **Remote Reference Module**
  - Translasi antara lokal dan remote objek reference menggunakan *remote object table*
    - Berisi data semua remote object dan proxy yang dipelihara oleh server
    - Digunakan ketika marshaling dan unmarshaling remote object reference
- RMI Sublayer terdiri dari
  - **Proxy** (di client), tempat penyimpan lokal untuk remote objek
  - **Dispatcher** (di server), menerima request dan menggunakan methodID untuk memilih message di skeleton
  - **Skeleton** (di server), menerapkan method dalam remote interface
    - Argumen-argument unmarshaling
    - Memanggil method yang sesuai pada remote objek
    - Menunggu selesainya pemanggilan
    - Melakukan marshaling hasil

Komponen RMI (2)

- **threads in server**
  - Setiap pemanggilan suatu method remote biasanya diberikan satu thread tersendiri untuk menjalankan method tersebut
- **Activation of remote object**
  - Mengaktifkan remote objek jika diperlukan saja
  - CORBA : Repository
- **Persistent Object store**
  - Objek remote dijamin dalam keadaan *live. Tapi pengaktifannya* tergantung request, dan jika sudah tidak dibutuhkan objek remote tersebut masuk dalam keadaan pasif.
  - Membutuhkan layanan Persistent Object (CORBA)

Komponen RMI (3)

- **factory methods**
  - Fakta: antarmuka objek remote tidak dapat memasukkan constructor. Program server berisi class untuk dispatcher dan skeleton bersama dengan class remote objek yang didukung (disebut dengan *servant).*
    - Program server ada bagian inisialisasi yang bertanggung jawab dalam pembuatan remote objek di server, atau
    - Pembuatan remote objek di server di dasarkan pada request dari client.
    - Pembuatan remote objek ini juga sering dilakukan proses binding.
  - Untuk pembuatan remote objek berdasar request dari user, pada antarmuka objek remote harus ada definisi *factory method*
- **Binders**
  - Layanan yang memelihara pemetaan dari kontek nama objek ke remote object references



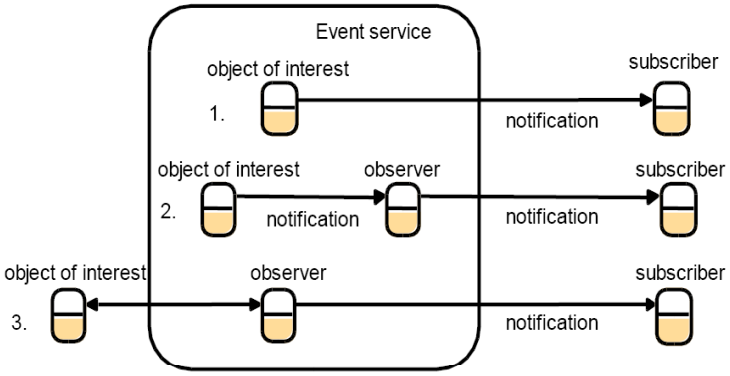
Distributed Garbage Collection

- **Algoritma :**
  - Tiap server memelihara daftar proses yang menyimpan referensi objek remote untuk setiap objek remote yang digunakan
  - Ketika client menerima sebuah referensi remote ke objek remote X, memanggil addRef(X) ke server, dan server menambah X ke daftar di server
  - Ketika garbage collector client diberi informasi oleh proxy bahwa objek X tidak diperlukan lagi, akan dikirim removeRef(X) ke server, dan server menghapus X dari daftar
  - Ketik daftar kosong, dan tidak ada referensi lokal ke X, kumpulkan resource yang digunakan X melalui local garbage collector server

Event dan Notification

- Event disebabkan adanya pemanggilan method atau pengiriman message
- Event mengubah status objek
- Beberapa remote objek di lokasi yang berbeda dapat diinformasikan suatu event pada suatu objek lain secara bersamaan, contoh :
  - Dokumen yang berubah
  - Seorang client baru bergabung dengan group
- Menerapkan paradigma *publish-subscribe*
  - Objek yang menghasilkan *event, mempublishkan tipe event* yang akan tersedia untuk diamati oleh objek lain
  - Objek yang ingin menerima notification dari objek yang mempublishkan eventnya, melakukan *subscribe ke tipe event* yang diinginkan
- Karakteristik :
  - Heterogen
  - asynchronicity

Arsitektur Event Notification



- Contoh :
  - CORBA Event Service
  - Java JINI Distributed Event