Programación Orientada a Objetos con Cocoa

Objetos Distribuidos

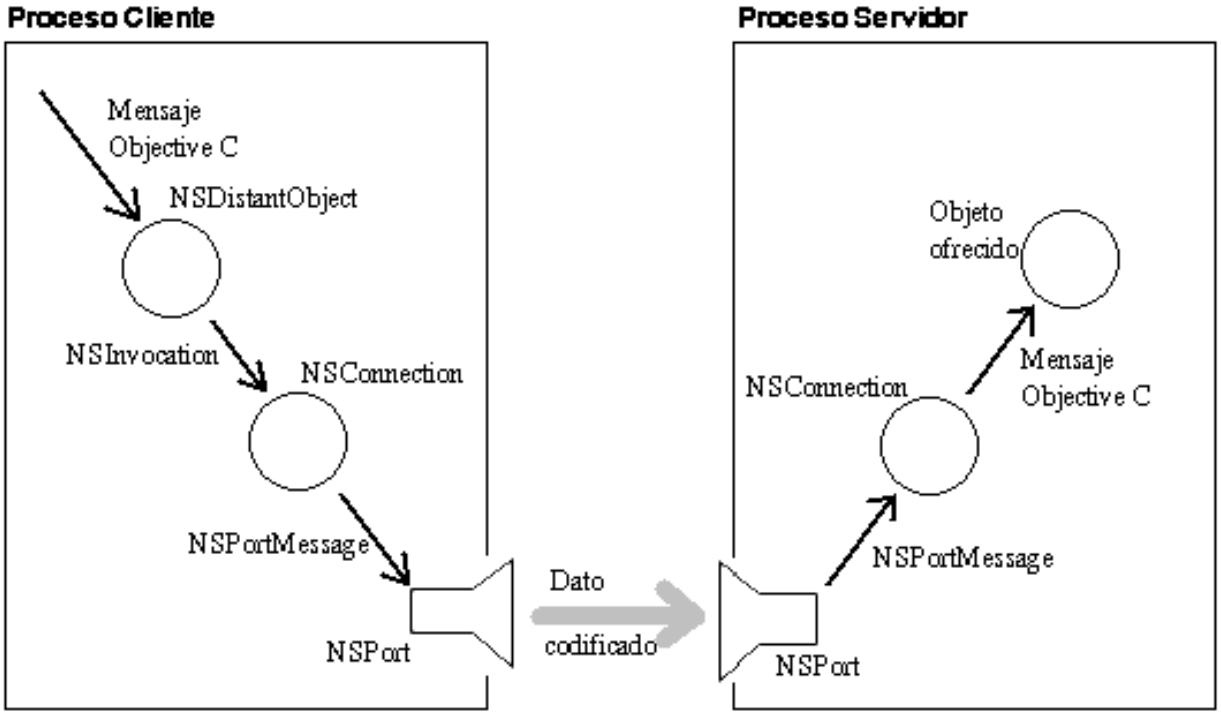
Amilcar Meneses Viveros

Sección Computación
Departamento de Ingeniería Eléctrica
CINVESTAV-IPN México, D.F.
(2003)

Objetos distribuidos

- Comunicación entre hilos, procesos o aplicaciones que pueden estar en diferentes máquinas.
- Distribuir aplicaciones
- Paralelizar tareas
- Arquitectura del RunTimeSystem
- Extensiones a Objective-C
- Sólo para aplicaciones Cocoa en Objective-C.

Arquitectura de los objetos distribuidos



Clases involucradas

`NSConnection` Maneja la comunicación entre objetos. Se utiliza para hacer disponible un objeto (en la red)

`NSProxy` Los objetos de esta clase se utilizan como representantes de objetos remotos.

`NSDistantObject` Es una subclase de `NSProxy`, y representa el objeto ofrecido del lado del cliente. Captura los mensajes y los manda al servidor a través de los objetos `NSConnection`.

`NSProtocolChecker` Es una sunclase de `NSProxy`, y se encarga de filtrar los mensajes (en el lado del servidor) al objeto distribuido "real".

Soporte del lenguaje

- Calificadores para el manejo de mensajes remotos, por cuestiones de rendimiento
- Por omisión: los mensajes son síncronos; los argumentos tipo apuntador se copian en el receptor y el emisor; y los objetos argumento se pasan por referencia a través de un objeto *proxy*.
- Calificadores:

  `oneway` Habilita un mensaje como asíncrono. Note que sólo con `void` tiene sentido.

  ```
  -(oneway void)mensaje;
  ```

  `in` Permite que un apuntador copie su contenido al objeto receptor.

  ```
  -(void)enviaDatoPorApuntador:(in tipoDato *)unApuntador;
  ```

Soporte del lenguaje...

- Calificadores:

  `out` Permite que un apuntador se le copie el valor de regreso al objeto emisor.

  ```
  -(void)recibeDatoPorApuntador:(out tipoDato *)unApuntador;
  ```

  ¡¿cómo sería el caso en que el mensaje regrese un valor tipo apuntador?!

  `inout` Se copian los valores a los objetos receptor y emisor.

  ```
  -(void)porOmision:(inout tipoDato *)unApuntador;
  ```

  Este modo consume mucho tiempo, es responsabilidad del programador verificar si se puede definir el mensaje del tipo `in` u `out`.

Soporte del lenguaje...

- Calificadores:

  `bycopy` Realiza una copia de un objeto argumento al receptor o al emisor

  ```
  -(void)mandaUnObjetoArgumento:(bycopy id)unObjeto;
  ```

  o bien, recibe un objeto por copia (¡¡no necesita `retain`!!)

  ```
  -(bycopy id)recibeObjeto;
  ```

  finalmente, combinado con `out`:

  ```
  -(void)recibeObjeto:(bycopy out id *)unObjeto;
  ```

  ¡¡¡Cuidado!!!, cuando se copie un objeto en este contexto, la aplicación que receptora **DEBE** tener la clase del objeto que se le manda.

  `byref` Paso de objetos argumentos por referencia. La referencia es a través de un *proxy*.

Ofreciendo un objeto

Para hacer disponible un objeto a otras aplicaciones, se hacen dos pasos: se pone como una raíz de `NSConnection` y después, se registra en la red:

```
id serverObject;    /* Asuma que existe */
NSConnection *theConnection;

theConnection=[NSConnection defaultConnection];
[theConnection setRootObject: serverObject];
if([theConnection registerName:@"MiServidor"]==NO){
    /* manejador de error */
}
```

Obteniendo un objeto disponible

Cuando se obtiene un objeto ofrecido por alguna aplicación, obtenemos un objeto representante (*"proxy"*).

```
id miProxy;
miProxy=[[NSConnection
      rootProxyForConnectionWithRegisteredName:@"MiServidor"
      host: nil] retain];
[miProxy setProtocolForProxy:@protocol(ServerProtocol)];
```

Otra forma es:

```
NSConnection *miConeccion;
id miProxy;
miConeccion=[NSConnection connectionWithRegisteredName:@"Miservidor"
                           host:nil];
miProxy=[[miConeccion rootProxy] retain];
[miProxy setProtocolForProxy:@protocol(ServerProtocol)];
```

Bibliografía

- Apple Documentation; *"Distributed Object"*;
  `http://developer.apple.com/documentation/Cocoa/`
  `Conceptual/DistrObjects/index.html`;
  Apple Inc.; April. 2003
- Amilcar Meneses & E. Enrique Martinez; Reporte del curso de programación orientada a objetos: *"Aplicación* Ticket *(Flecha-Rota), un ejemplo del uso de objetos distribuidos"*; Sección de Computacion, CINVESTAV-IPN, Dic. 1998.