Programación Concurrente y Distribuida

Ingeniería Informática
Facultad de Ciencias
Universidad de Cantabria.

Documento:

Aplicación SmartHunter con ICE

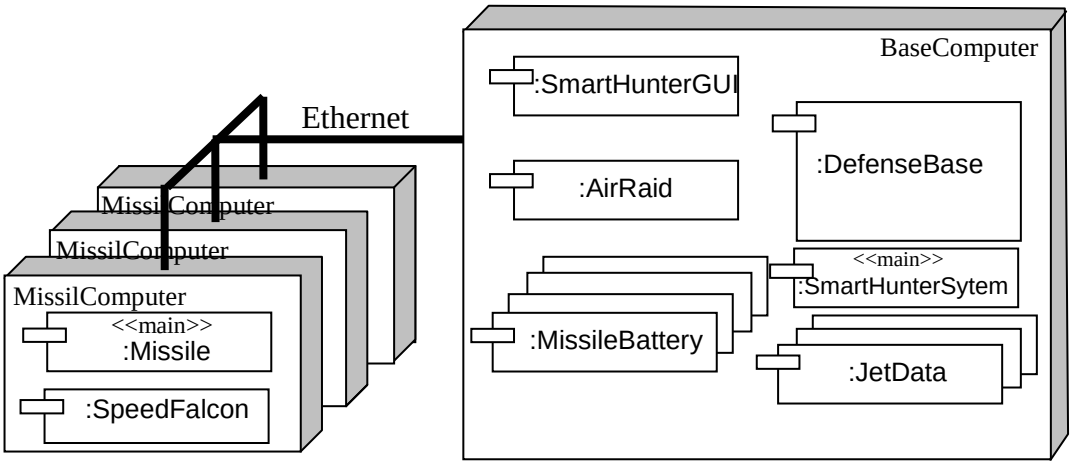
Autores: Laura Barros
Miguel Tellería
JM Drake

Santander, diciembre, 2011

Distribución de la aplicación SmartHunter con Ice.

Opción del Miércoles

En este caso se propone la distribución de la aplicación utilizando Ice, de acuerdo con el siguiente plan de despliegue:



Se considera que todo el software del sistema de defensa está instalado en un único computador, mientras que se distribuye en cada misil, tanto el software de control del hardware (SpeedFalcon), como el software de conducción del misil (Missile).

Criterios de diseño

La distribución afecta a cada misil, esto es, los objetos relativos a cada misil Missile y SpeedFalcon, quedan instanciados en el procesador MissileComputer, mientras que las clases de los servicios comunes se instancian en el procesador BaseComputer. En este caso las instancias Missile actuan como clientes, mientras que las clases MissileBattery actúan como servidores.

Hay que resolver la forma en que las instancias Missile y MissilBattery obtienen el puerto a través de la que interactúan, y la posición inicial desde la que se dispara:

- Los objetos de la clase MissileBattery tiene asignados de forma estática los puertos a través de los que atiende:
    - Battery “E” [10000,0] => 3000
    - Battery “N” [0,10000] => 3001
    - Battery “S” [-10000,0] => 3002
    - Battery “O” [0,-10000] => 3003

  Cuando las baterías se instancian, se registran en su adaptador de objetos con el nombre de la batería “E”,“N”,”S” o “O”.
- Cuando los programas Missile se ejecutan, reciben como argumento de su main() el nombre de la batería a la que se asigna, esto es “E”,“N”,”S” o “O”, y con él crea el proxy para el acceso a la batería.

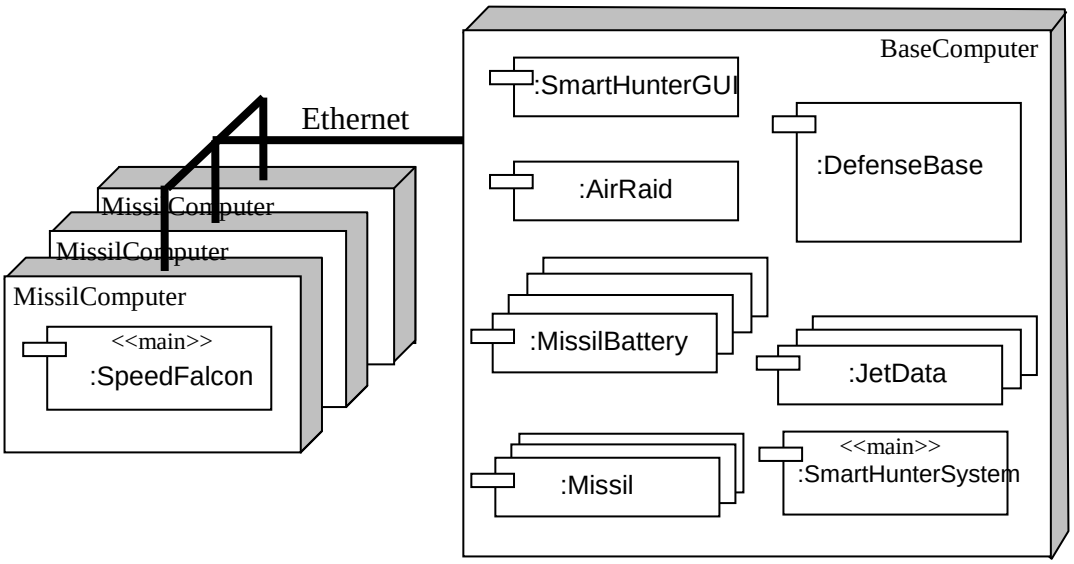
Nota: *En esta práctica hay que modificar:*

- *la clase **MissileBattery**: Se debe convertir en un servidor Ice que atienda a los misiles.*
- *La clase **Missile** que se convierte en un programa con su main() y que se constituye en un cliente del servidor ofrecido por las instancias de MissileBattery.*
- *El programa principal **SmartHunterSystem** que no debe instanciar los misiles.*

Distribución de la aplicación SmartHunter con Ice.

Opción del Jueves

En este caso se propone la distribución de la aplicación utilizando Ice, de acuerdo con el siguiente plan de despliegue:



Se considera que todo el software del sistema de defensa está instalado en un único computador, mientras que se distribuye en cada misil sólo el software de control el hardware.

Criterios de diseños

La distribución afecta a cada misil, esto es, los objetos de la clase misil quedan instanciados en las clases MissileBattery sobre el procesador BaseComputer, mientras que las instancias SpeedFalcon son instanciadas en los procesadores MissileComputer. Las instancias MissileComputer actúan como servidores, mientras que las instancias Missile actúan como clientes.

Hay que resolver la forma en que las instancias Missile y SpeedFalcon obtienen el puerto a través de la que interactúan, y la posición inicial desde la que se dispara:

- Los Programas de la clase SpeedFalcon se registran en el adaptador de objetos con nombres compuestos del nombre de la batería y un ordinal, y para construir el adaptador, utilizan un puerto definido a partir de una base propia de la batería en la que se instala:
    - Battery “E” => Puertos 3000,3001,3002, ... y con nombres “E1”,”E2”,”E3”,...
    - Battery “N” => Puertos 3100,3101,3102, ... y con nombres “N1”,”N2”,”N3”,...
    - Battery “S” => Puertos 3200,3201,3202, ... y con nombres “S1”,”S2”,”S3”,...
    - Battery “O” => Puertos 3300,3301,3302, ... y con nombres “O1”,”O2”,”O3”,...

- Los objetos Missile cuando son creados por la MissileBattery, reciben en su constructor el nombre de la batería a la que son asignados, y en función de él, construyen un Proxy de algún SpeedFalcon que esté registrado con los nombres que corresponden a su batería: “N0”, “S8”, “E2”...

Nota: *En esta práctica hay que modificar:*

- *La clase MissileBattery que debe crear el objeto Missile en su método shot.*
- *La clase **Missile**: Se debe convertir en un cliente Ice que localiza el SpeedFalcon que controla..*
- *La clase **SpeedFalcon** se constituye en un servidor Ice que atiende a su controlador Missile.*
- *El programa principal **SmartHunterSystem** no debe instanciar los misiles.*