R-Grid: Un Grid Reconfigurable basado en FPGA para Computación de Alto Rendimiento

F. Sánchez, F. Rincón, F. Moya, J.C. López, J. Barba, J. Dondo

E.S. Ingeniería Informática

Univ. Castilla-La Mancha

fco.sanchez5@alu.uclm.es{ fernando.rincon, francisco.moya, juancarlos.lopez, Jesus.Barba, jdondo } @uclm.es

Resumen

Uno de los campos de utilización de las FPGAs en los últimos años ha sido la computación de alto rendimiento. Esta tecnología ofrece mejoras de varios ordenes de magnitud en cuanto a prestaciones, pero su elevado coste y la complejidad de la programación impiden una mayor implantación. Por el contrario, los sistemas basados en grid se han desarrollado y extendido ampliamente, convirtiéndose en la propuesta de mayor aceptación para computación de alto rendimiento. La aportación del grid es una arquitectura que abstrae los recursos, su aprovechamiento, el despliegue de aplicaciones y el sistema de comunicación. En este artículo se propondrá una arquitectura destinada a FPGAs que integre soluciones grid, y que permita aprovechar las nuevas tendencias y avances tecnológicos en la electrónica reconfigurable.

1. Introducción

A través de los diferentes estudios en el área de la electrónica reconfigurable se ha llegado a la conclusión de la viabilidad de la utilización de FPGAs en aplicaciones de alto rendimiento. Como puntos fuertes las FPGAs ofrecen un rendimiento de varios ordenes de magnitud superior al de un procesador de propósito general, la posibilidad de adaptarse a la mayoría de problemas paralelos y por último un consumo de energía muy contenido. Pero si estas plataformas no terminan de despegar se debe principalmente a dos graves inconvenientes, por un lado su elevado precio actual con un ratio rendimiento / coste peor que el de otras soluciones [2], y por otro lado, y no menos importante, la complejidad del desarrollo de aplicaciones para estas plataformas.

El elevado precio depende en gran medida de la cuota de mercado. Una mayor tasa de implantación repercutiría directamente en los precios, provocando una caída de estos. Como el inconveniente económico sólo se puede resolver si se aumenta la competitividad del producto con respecto a otras tecnologías, queda claro que hay que atacar la complejidad de programación para que ésta no sea el obstáculo respecto a computadores convencionales o GPUs.

La investigación basada en el modelo de programación orientada a objetos en el ámbito de la electrónica reconfigurable, ofrece al desarrollador de FPGAs servicios y prestaciones de sistemas distribuidos, como la transparencia de localización, reutilización sencilla gracias a la definición de interfaces de forma sistemática, y facilidades a la hora de plantear la arquitectura de intercomunicación [9]. Estos servicios están orientados a crear una capa de abstracción que permita aislar al desarrollador de sistemas de la tecnología destino, y a facilitar el co-diseño gracias a las facilidades de intercomunicación HW-SW. Como resultado de esta línea de investigación, el grupo ArCo desarrolló un middleware de comunicaciones distribuido hardware denominado OOCE [1].

En software, los sistemas distribuidos han evolucionado aportando cada vez más servicios y prestaciones, derivando en sistemas clusters

y grids. Las soluciones que aportan estos últimos, también resuelven problemas análogos a los que aparecen en sistemas empotrados, como el despliegue de aplicaciones, dependencia de la tecnología, o dependencia de la topología. Los sistemas cluster y grid proponen modelos que permiten tratar estos problemas de forma sencilla y homogénea. Sistemas distribuidos clásicos como Ice con la inclusión de IceGrid, también han evolucionado hacia la inclusión de estos servicios [7].

Estas soluciones desarrolladas ampliamente en software, se han convertido en una de las técnicas con mayor relevancia a la hora de cubrir las necesidades de cómputo de alto rendimiento en aplicaciones con un marcado carácter científico [6, 4, 5].

Partiendo del middleware OOCE y de que los sistemas clusters y grids se enfrentan a problemas parecidos al de sistemas empotrados, plantearemos una línea de investigación que se basa en aplicar los conceptos de los sistemas grids en electrónica reconfigurable. Esta visión de sistema nos ofrece un nuevo campo en el que ofreceremos una serie de servicios que facilitarán el desarrollo y despliegue, creando nuevos modelos de explotación de FPGAs.

En resumen en este artículo desarrollaremos una arquitectura para tratar los problemas del diseño, despliegue, rendimiento y seguridad de sistemas de alto rendimiento basados en FPGAs, planteando una analogía con los sistemas cluster y grid. Esta arquitectura, a la que denominaremos "Reconfigurable-Grid" o simplemente "R-Grid", pretende aportar transparencia tanto en la tecnología que se utilice como en la topología del sistema y su ubicación.

En la siguiente sección trataremos el "Estado del arte", dónde plasmaremos por un lado la situación actual en los sistemas empotrados orientados al alto rendimiento, y por otro las soluciones grid utilizadas en el mundo software. A continuación, en la sección "Objetivos", nos fijaremos las metas que deseamos alcanzar con la arquitectura "R-Grid" en cuanto a facilidad de uso, rendimiento y seguridad. En la sección "Arquitectura R-Grid" desarrollaremos un modelo que nos permita ofrecer los servicios necesarios para cumplir todos los objetivos planteados. Después en el apartado "Configuración y despliegue" se describirá cómo interactuan los componentes de la arquitectura con la ayuda de un ejemplo del ciclo de vida de una aplicación. La sección "Resultados experimentales" nos dará una idea de los recursos que utiliza la arquitectura y el área disponible para el usuario. Por último en las "Conclusiones" se resumirán los objetivos alcanzados.

2. Estado del arte

El objetivo en los sistemas de alto rendimiento es simple, llegar al equilibrio entre las máximas prestaciones, el menor coste y consumo, garantizando un mínimo de rendimiento. Estudios como [2] demuestran como las FPGAs ofrecen cada vez mayores prestaciones a un precio más competitivo, y además dicha progresión es mayor en FPGAs que en sistemas de procesadores de propósito general. Este factor permitirá una mayor implantación de la electrónica reconfigurable en computadores de alto rendimiento.

A pesar de que actualmente el coste de las FPGAs sigue siendo mayor, diferentes fabricantes de computadores de alto rendimiento como SGI, SRC Computers o Cray ya han creado arquitecturas en las que combinan procesadores de propósito general y FPGAs, interconectando ambos elementos de cómputo mediante buses de alta capacidad. Estas máquinas se conocen como High Performance Reconfigurable Computers (HPRCs) [3]. Su modelo se basa en ofrecer facilidades en forma de biblioteca software que oculta parte del hardware, permitiendo mediante llamadas software al sistema la carga del bitstream y su ejecución, convirtiendo así a las FPGAs en coprocesadores para resolver las partes algorítmicas con mayor peso.

La inclusión de FPGAs en computadores de alto rendimiento destinados a formar sistemas clusters y grids, nos hace pensar en una arquitectura que permita tratar las FPGAs como el resto de elementos del sistema. Puesto que la visión de una FPGA como un simple elemento esclavo, provoca que la aceleración sea limitada debido al intercambio constante de datos

entre la zona software y la hardware.

Acercándonos al mundo software relacionado con la computación grid, descubrimos un debate sobre lo que se considera estándar en cuanto al modelo de comunicaciones. A pesar del debate, una gran cantidad de usuarios y aplicaciones grid ya han optado por utilizar "Web services" [8]. Este modelo incluye, en principio, mejoras en el ámbito de la interoperabilidad y la escalabilidad. Los "Web Services" aportan una arquitectura en la que mediante un lenguaje de descripción se define la interfaz (WSDL) y después se elige el canal de transmisión de los mensajes, lo que aporta servicios ligeros y con bajo acoplamiento.

La sobrecarga que añadiría el modelo de comunicación "Web Services" en hardware tal y como podemos ver que la provoca en un nodo software de un grid [10, 11], nos hace descartar dicho modelo debido a que choca frontalmente con el objetivo de alto rendimiento. En hardware, no son admisibles los tiempos de respuesta necesarios para procesar las complejas invocaciones realizadas mediante tecnología "WebService". Pero no por ello debemos olvidar la posibilidad de integrarlo mediante un módulo externo que utilice la arquitectura, convirtiéndose en otro punto a explorar.

3. Objetivos

El proyecto R-Grid pretende crear una plataforma hardware de computación de alto rendimiento distribuida, orientada a la explotación concurrente y descentralizada de racimos heterogéneos de FPGAs. Los mecanismos de gestión que se propondrán en este trabajo proporcionan servicios que guardan cierta analogía con los ofrecidos por un grid de computadores, o por arquitecturas de objetos distribuidos.

El proyecto abarca el desarrollo de un modelo de componentes hardware basado en un modelo de sistemas de objetos distribuidos fruto de proyectos anteriores. Además se propone el estudio de estrategias que complementen el modelo de comunicación entre objetos para permitir toda una gama de servicios avanzados de la plataforma (despliegue y configuración remota, replicación, migración, seguridad, etc.).

Las posibles aplicaciones abarcan problemas altamente paralelizables que no se ajustan al modelo actual de computación en grid debido a latencias excesivas o rendimiento (throughput) insuficiente. También posibilita nuevos modelos de negocio mediante la externalización de las plataformas de desarrollo hardware.

A continuación se detallan los objetivos que se propone cubrir la plataforma R-Grid para dar soporte al desarrollo y explotación de sistemas de FPGAs. Estos objetivos se dividen según su finalidad, dependiendo de si están destinados a facilitar el uso del sistema, generar la arquitectura de alto rendimiento, o dotar a la plataforma de medidas de seguridad.

3.1. Facilidad de uso

La facilidad de uso pretende que el usuario pueda interactuar con el sistema a través de una interfaz bien definida que oculte los detalles tecnológicos concretos.

- **Despliegue:** Servicio para cargar las aplicaciones en las FPGAs, independiente de la arquitectura concreta.
- **Transparencia:** Mecanismo de localización autónomo para la invocación de objetos cargados de los que se desconozca su ubicación.
- **Replicación:** Servicio y modelo pensado para replicar objetos y ofrecer balance de carga.
- **Migración:** Servicio para detener una aplicación, almacenar su estado, y recuperarlo para continuar la tarea.

3.2. Rendimiento

No hay que olvidar que nos centramos en aplicaciones de alto rendimiento.

- **Eficiencia:** La arquitectura no representará una carga en los recursos del sistema, sus servicios han de ser ligeros y rápidos.

- **Rendimiento:** El sistema de comunicación tendrá alta capacidad y baja latencia.

3.3. Seguridad

Los elementos de seguridad siempre son necesarios en sistemas distribuidos. R-Grid no va a ser una excepción, aportando al usuario un entorno confiable en el que no se ponga en riesgo la privacidad.

- **Secreto:** Los datos con los que trabaja una aplicación no pueden ser revelados a terceros.
- **Integridad:** Los datos de una aplicación no se pondrán alterar por terceros.
- **Disponibilidad:** Una aplicación mal intencionada no podrá inutilizar ni el acceso, ni la funcionalidad del sistema.

4. Arquitectura R-Grid

La plataforma R-Grid coordina multitud de FPGAs, recursos y áreas programables para la ejecución de aplicaciones distribuidas de alto rendimiento. Su tarea principal se basa en realizar una partición eficiente de los recursos de cada FPGA y administrarlos para ofrecerlos al usuario. Dicha tarea implica la puesta en marcha de servicios de despliegue y localización transparente, seguridad, o recursos de memoria entre otros. Para ofrecer estos servicios, dividiremos las responsabilidades en diversos componentes.

Comenzaremos así por definir, tal como vemos en la figura 1, un coordinador general responsable de la interacción con el usuario y de la utilización de los servicios de bajo nivel. Por otro lado, dentro de cada FPGA definiremos una parte estática que se encargará de los servicios de reconfiguración, localización y seguridad, y otra parte dinámica, que albergará zonas de recursos donde se instanciarán los bitstreams del usuario.

Los tres servicios que se ofrecen en la parte estática del hardware se repartirán en tres

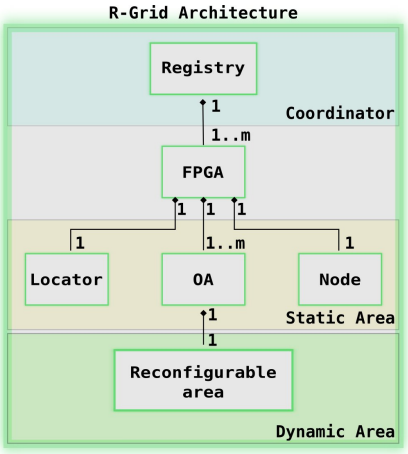

Figura 1: Diagrama de clases del sistema R-Grid

componentes, "Node" se encargará de la reconfiguración, "Locator" de la localización y "Object Adapter" de la seguridad. Estos componentes crean la capa de abstracción tecnológica para las aplicaciones, son puramente hardware y dependen de la tecnología de implementación. Además de la división de estas tres funcionalidades, en la zona estática también se localiza la arquitectura de intercomunicación, que será utilizada por las aplicaciones que se carguen en el sistema.

A continuación se detalla la funcionalidad de cada componente.

4.1. Registry

El objeto "Registry" será único en el sistema. Su función comienza con el registro de descriptores de aplicación en memoria. Estos descriptores contienen la información básica de la aplicación, su nombre, los bitstreams que contiene y los objetos que se desplegarán con sus direcciones locales.

Para detallar su funcionalidad se muestra el código de usuario necesario para completar el ciclo de vida de una aplicación en "R-Grid" utilizando la interfaz de "Registry":

```
RegistryPrx r = new RegistryPrx();
Descriptor d =
  new Descriptor("App1",
```

```
                    ["serverName",
                    "/bitstream/path",
                    ["obj1",0x00001010
                        ]]);
r.addDescriptor(d);
r.startServer("App1","serverName");
Obj1Prx obj1 = new Obj1Prx();
obj1.hello();
r.stopServer("App1","serverName");
r.removeDescriptor("App1");
```

Algoritmo 1: Ciclo de vida de una aplicación

En el ejemplo el usuario en primer lugar crea un "Proxy" de "Registry" y un descriptor de la aplicación que pretende desplegar. El descriptor contiene los datos del nombre de la aplicación "App1", la cual solo alberga un servidor o bitstream parcial de nombre "serverName", localizado en "/bitsteam/path", y que contiene un sólo objeto de nombre "obj1" en la dirección local "0x00001010". Después registra el descriptor en "Registry", le pide que inicie el servidor "serverName" de la aplicación "App1" y ejecuta el método "hello" del objeto. Por último detiene el servidor y elimina el descriptor de aplicación.

Los métodos "startServer" y "stopServer" desencadenan peticiones de "Registry" hacia servicios que ofrecen los componentes "Node", "Locator" y "Object Adapter", con el objetivo de instanciar los bitstreams y registrar los objetos cargados. Además se encargará de la persistencia de los objetos que así lo necesiten, ofreciendo la memoria necesaria para que almacenen su estado. Los datos almacenados se ofrecerán en el momento que se vuelva a instanciar el objeto persistente.

4.2. Zona estática

La zona estática es la parte de la FPGA destinada a los componentes hardware "Locator", "Object Adapter" y "Node". En la figura de ejemplo 2 podemos ver cómo se distribuyen e interconectan.

4.2.1. Node

El objeto "Node" permitirá abstraer al sistema "R-Grid" del modelo específico de reconfiguración parcial que cada FPGA utilice, del número de recursos de programación libres y del lugar donde éstos se ubiquen. Es decir, "Registry" tendrá la visión de una interfaz que invocará con independencia del modelo concreto de la FPGA en cuestión.

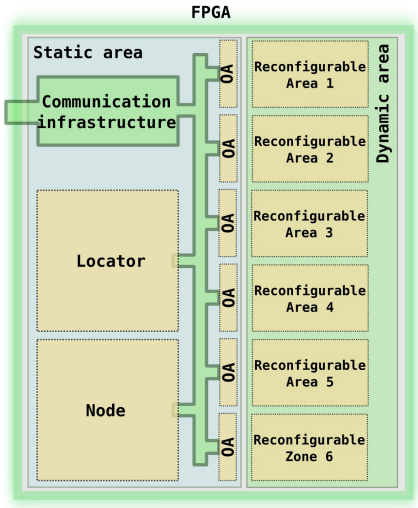

Figura 2: Diagrama de bloques en la FPGA

Cada FPGA que forme parte del sistema "R-Grid" contendrá un objeto "Node" como se puede observar en la figura 2, que se encargará de mantener la información de recursos disponibles actualizada, y de programar bitstreams en áreas libres [9].

A nivel de implementación "Node" utilizará de forma directa el dispositivo "Icap" que contienen las FPGA de la familia "Virtex". De esta forma, "Node" es un elemento autónomo que permite realizar la programación parcial de bitstreams.

A continuación un ejemplo de cómo instanciar y detener un bitstream con la ayuda del objeto "Node", pasos que realizará el componente "Registry".

```
NodePrx n = NodePrx();
int d = n.startServer ("/bitstream/
    path");
n.stopServer (d);
```

Algoritmo 2: Utilización de "Node"

4.2.2. Locator

"R-Grid" aportará acceso transparente a los recursos, permitiendo invocar métodos de objetos a partir de su nombre sin necesidad de conocer su localización exacta, y sin intervención de la aplicación del usuario. Esta transparencia se consigue mediante el servicio que ofrece el objeto "Locator", encargado de la localización de objetos en el sistema en tiempo de ejecución.

La transparencia de localización involucra, además de "Locator", a los objetos "proxy" que son una representación del objeto al que se invoca dentro del cliente. Dichos "proxies" son los responsables en última instancia de realizar la invocación en el sistema, y por tanto, responsables de asignar la dirección de la invocación. Si ésta es desconocida cuando se invoca un método, la podrá obtener mediante el servicio de localización, de forma transparente al usuario [9].

Como se ha comentado anteriormente, "Registry" se encargará de indicar a "Locator" los nuevos objetos que se instancien en el sistema, o los que han dejado de estar disponibles. "Locator", gracias a esta información, mantendrá actualizada una memoria con todos los objetos disponibles en la FPGA.

El servicio de localización podría ser único y estar ubicado en "Registry", pero por razones de eficiencia, se dispondrá de un "Locator" por FPGA para que las consultas locales se resuelvan lo más rápido posible, pero sin perder búsquedas globales en el sistema mediante consultas jerárquicas.

En las siguientes líneas se muestra un ejemplo de registro y búsqueda en el objeto "Locator".

```
LocatorPrx l = LocatorPrx();
l.registerObject("Obj1", 0x4F002000)
    ;
l.locate("Obj1");
>>> 0x4F002000
l.unregisterObject("Obj1");
```

Algoritmo 3: Utilización de "Locator"

4.2.3. Object Adapter

Su tarea se basa en controlar la comunicación que realiza una aplicación cargada con el sistema. Cada "Object Adapter" contará con una dirección global en el sistema, que será el primer filtro para que una invocación llegue al área reconfigurable. Por otro lado, cuenta con listas de control de acceso tal y como podemos ver en la figura 3, que permiten decidir si una invocación del área reconfigurable es legal. Este control permite dotar al sistema R-Grid de las medidas de seguridad necesarias para una ejecución aislada de aplicaciones.

Esta configuración la realizará el objeto "Registry", que contará con una vista de administración de "Object Adapter".

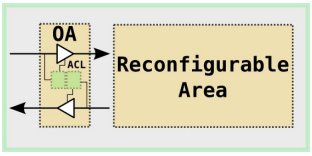

Figura 3: Detalle de "Object Adapter"

4.3. Área dinámica

El área dinámica es la zona reservada en la FPGA para albergar los bitstreams o servidores del usuario. Cada servidor contiene una o varias funcionalidades de la aplicación, y está compuesto por el número de objetos que el desarrollador estime oportuno, con la limitación del tamaño del área reconfigurable.En el modelo que se ha considerado inicialmente las áreas cuentan con un tamaño razonable y todas son iguales. Esta división permite la reubicación dinámica de bitstreams por parte del objeto "Node".

5. Configuración y despliegue

Este apartado mostrará las fases de diseño, desarrollo, despliegue y ejecución de una aplicación basada en la arquitectura "R-Grid".

El primer paso será diseñar la aplicación teniendo en cuenta el modelo de desarrollo de objetos distribuidos.

Por otro lado, mediante el middleware OOCE, se definirán las interfaces de los objetos, y herramientas automáticas crearan "proxies" y "skeletons" a partir de ellas.

En esta fase en la que ya tenemos los objetos que forman parte de la aplicación, su tamaño y como se comunican entre ellos, se realizará una partición eficiente para situarlos en bitstreams parciales. Intentado agrupar los objetos con mayor comunicación entre ellos.

La siguiente etapa comienza con la descripción de la aplicación. Entre los datos que forman parte de la descripción se encuentran el nombre de la aplicación, los bitstreams que implica, y los objetos con su dirección local que se definen dentro de cada bitstream parcial.

Los siguientes pasos están detallados en la figura 4, que comienza con almacenar en "Registry" el descriptor, y el posterior arranque de la aplicación en el paso 1. Después en el paso 2 el sistema carga el bitstream mediante "Node", se realizan las tareas necesarias para la persistencia en el paso 3, y en 4 se registran los objetos instanciados en "Locator", además de configurar el adaptador de objetos.

En el paso 5, con la aplicación desplegada y lista para utilizarse, se ejemplifica la localización de un objeto, y su posterior invocación.

En el paso 6 se realiza la secuencia de detención de la aplicación, marcada en 7 con la configuración del adaptador para incomunicar al servidor y la eliminación de los objetos registrados en "Locator" pertenecientes a la aplicación que se detiene. Se almacena el estado y en 8 se detiene definitivamente el servidor. Para concluir el ciclo de vida, en 9 se elimina el descriptor de aplicación.

6. Resultados experimentales

La arquitectura planteada para "R-Grid" está en fase de desarrollo, encontrándose actualmente a nivel experimental. En esta fase se cubren de forma básica los servicios de los objetos "Node" y "Locator", dando una idea aproximada del tamaño que puede representar cada componente en un dispositivo concreto, así como del tamaño y número de áreas reconfigurables que se pueden tratar.

La implementación se está realizando sobre una placa de prototipado Xilinx Virtex-4 XC4VLX60 y en un SGI Altix 450.

En el cuadro 1 podemos encontrar un resumen con las características de cada objeto implementado en la placa Virtex-4 XC4VLX60.

Component	Slices	Slices FFs	LUTs	IOs
Locator	281	159	528	104
Node	190	167	359	211

Cuadro 1: Utilización de recursos

En las pruebas realizadas con el dispositivo Virtex-4 XC4VLX60, se ha conseguido implementar la arquitectura y programar ocho áreas reconfigurables con 1040 slices cada una. La elección del número de áreas reconfigurables y su tamaño se debe a la arquitectura de la FPGA y las restricciones que aparecen en las áreas reconfigurables. La restricción que más se ha tenido en cuenta ha sido la de que una zona reconfigurable debe quedar contenida en un dominio de reloj local. De la Virtex-4 XC-VLX60 se ha destinado media FPGA para el área dinámica, en la que existen 8 zonas de reloj local. La utilización de la mitad de la FPGA se debe a que el desarrollo está en fase experimental.

7. Conclusiones y Trabajo Futuro

Uno de los problemas de la utilización de FPGAs es la complejidad de su desarrollo. Por ello, el modelo de objetos distribuidos que adopta "R-Grid" facilita el desarrollo de aplicaciones y su posterior utilización. El uso de la plataforma también permite mantener los roles de desarrolladores separados: por un lado el desarrollador hardware, por otro el desarrollador software y por último el integrador.

Otro punto que facilita el desarrollo e implantación es que la arquitectura es adaptable a distintos tipos de tecnología reconfigurable, mejorando la portabilidad y el aprovechamiento de recursos.

Además, la arquitectura "R-Grid" se adapta mejor al uso de las FPGAs como aceleradores, consiguiendo que los nodos de computación

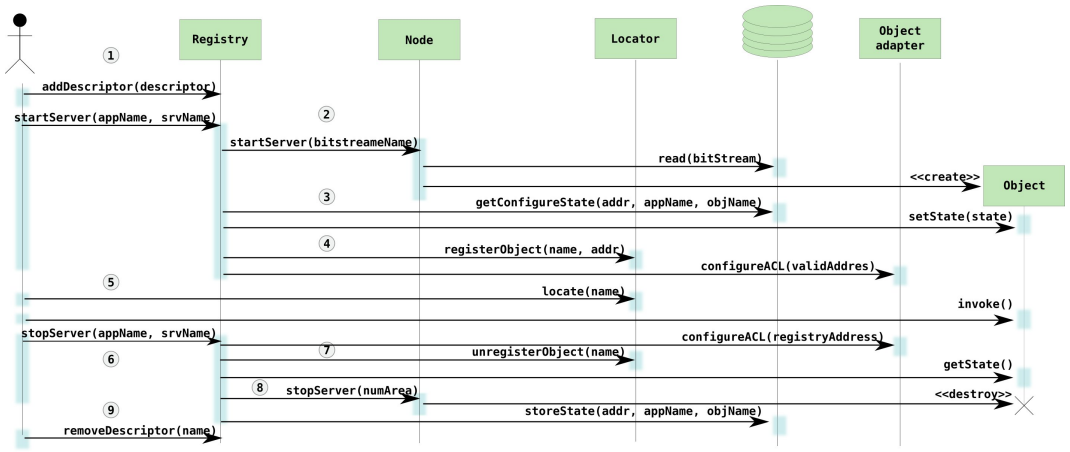

Figura 4: Diagrama de secuencia "R-Grid"

sean explotables por si mismos. Esto reduce el cuello de botella del control software.

Como trabajo futuro queda el desarrollo completo de los prototipos, y la creación de un caso de estudio para la caracterización de diferentes aspectos, tanto de rendimiento como de facilidad de uso.

Referencias

[1] J. Barba, F. Rincon, F. Moya, F.J. Villanueva, D. Villa, J. Dondo, and J.C. Lopez. Ooce: Object-oriented communication engine for soc design. In *DSD 2007.Euromicro*, aug. 2007.

[2] S. Craven and P. Athanas. Examining the viability of fpga supercomputing. *EURASIP*, 2007(1), 2007.

[3] E. El-Araby, I. Gonzalez, and T. El-Ghazawi. Performance bounds of partial run-time reconfiguration in hprc. In *HPRCTA '07*, pages 11–20. ACM, 2007.

[4] Nigel G. Fielding. Grid computing and qualitative social science. *Soc. Sci. Comput. Rev.*, 26(3):301–316, 2008.

[5] Bob Jones. The use of grids for fusion applications in europe and future directions. In *GMAC '09*, pages 39–40, New York, NY, USA, 2009. ACM.

[6] T. Kurc, S. Hastings, V. Kumar, and Langella. Hpc and grid computing for integrative biomedical research. *Int. J. High Perform. Comput. Appl.*, 23(3):252–264, 2009.

[7] M. Spruiell M. Henning. Distributed programming with ice, Revision 3.4, February 2010.

[8] Liang Peng, Simon See, Yueqin Jiang, Jie Song, Appie Stoelwinder, and Hoon Kang Neo. Performance evaluation in computational grid environments. *HPC and Grid in Asia Pacific Region*, 0, 2004.

[9] F. Rincón, J. Dondo, J. Barba, F. Moya, and J.C. López. Supporting operating systems for reconfigurable computing: A distributed service oriented approach. In *ERSA*, 2009.

[10] F. Taiani, M. Hiltunen, and R. Schlichting. The impact of web service integration on grid performance. In *HPDC 2005*.

[11] D. Villa, F. Jesús Villanueva, F. Moya, F: Rincón, J. Barba, and J.C. López. Web services for deeply embedded extra low-cost devices. In *GPC*, 2009.