

## **Mecanismo de distribución y ejecución de procesos utilizando Microsoft .NET**

Sebastián Baña  
sbana@ei.edu.uy

Gonzalo Ferres  
gferres@ei.edu.uy

Sergio Nesmachnow  
sergion@ei.edu.uy

Nicolás Pepe  
npepe@cs.com.uy

Universitario Autónomo del Sur  
Montevideo – Uruguay

Junio 2002

### **Resumen**

Este documento describe una propuesta de diseño e implementación de un mecanismo de distribución y ejecución remota de procesos en Internet, basado en la tecnología Microsoft .NET.

El mecanismo de distribución y ejecución es uno de los objetivos específicos del proyecto MPI.NET, que propone utilizar la tecnología Microsoft .NET para diseñar una implementación de la biblioteca de desarrollo de aplicaciones paralelas y distribuidas MPI (Message Passing Interface), capaz de ejecutar en Internet.

Palabras clave : programación paralela y distribuida, Microsoft .NET, distribución de procesos, ejecución de procesos, MPI.

## Mecanismo de distribución y ejecución de procesos utilizando Microsoft .NET

Este documento describe una propuesta de diseño e implementación de un mecanismo de distribución y ejecución remota de procesos en Internet, basado en la tecnología Microsoft .NET. El trabajo presentado se enmarca dentro del proyecto MPI.NET, que propone utilizar la tecnología Microsoft .NET para construir una versión de la biblioteca para programación paralela y distribuida MPI adecuada para el uso en Internet.

El sitio WEB del proyecto MPI.NET se encuentra en <http://www.npsistemas.com.uy/mpi.net/>

El documento se organiza de la siguiente manera:

En la sección 1 se brinda una introducción a la propuesta y el marco del proyecto global. La sección 2 reúne una revisión de antecedentes y conceptos relacionados al problema de la distribución de procesos en la programación distribuida y en la biblioteca MPI en particular, complementado con comentarios de trabajos existentes en el área. La sección 3 brinda una descripción detallada del problema, incluyendo los aspectos que las propuestas previas no resuelven y la importancia de los mismos. La propuesta de diseño y los comentarios sobre la implementación del prototipo del sistema de distribución se ofrecen en la sección 4. Las conclusiones y las ideas para trabajo futuro se describen en la sección 5.

### 1. Introducción

La biblioteca de programación paralela y distribuida MPI [1] permite operar un conjunto de máquinas interconectadas de forma unificada, como una máquina virtual paralela que coordina el trabajo de diferentes procesadores mediante primitivas accesibles desde los lenguajes C, C++ y FORTRAN.

Las implementaciones actuales de la biblioteca MPI no permiten trabajar utilizando equipos en diferentes LANs (Local Area Networks – Redes de Área Local), así como tampoco usar fácilmente la biblioteca desde otros lenguajes.

Avances sostenidos en la tecnología, especialmente el incremento en la capacidad de procesamiento, ancho de banda de red, y la disponibilidad de estándares de programación y comunicación han creado el entorno que ha hecho posible un nuevo ambiente de trabajo, la plataforma Microsoft.NET [2].

Este entorno toma ventaja de la adaptación de protocolos estándar de Internet, permitiendo el desarrollo de aplicaciones distribuidas multilenguaje y multiplataforma independientes o que utilicen servicios existentes a través de un modelo de programación popular y abierto.

En el marco de esta realidad, el proyecto MPI.NET propone utilizar la tecnología Microsoft .NET para el desarrollo de una implementación de MPI que toma como uno de sus principios fundamentales de diseño el uso de protocolos estándar de Internet.

De esta manera se pretende brindar una herramienta para el desarrollo de aplicaciones distribuidas en el entorno Microsoft .NET que sea adecuada para ejecutar en Internet, y posibilitar su utilización desde los diferentes lenguajes que tienen su implementación en Microsoft .NET.

El mecanismo de distribución y ejecución propuesto permite distribuir y ejecutar el código en los nodos de la máquina paralela virtual, los cuales pueden formar parte de diferentes redes.

### 2. Antecedentes

El mecanismo de distribución y ejecución de procesos debe proveer una primitiva que permita crear procesos asociándoles un código para su ejecución distribuida en Internet.

La funcionalidad de creación de procesos no fue incluida en el primer estándar MPI, quedando su definición en manos de los desarrolladores, quienes deberían proveer al usuario la forma de poder especificar en el momento de la ejecución la cantidad inicial de procesos, el código a ser ejecutado y su ubicación en relación a los procesadores disponibles en la máquina virtual [3].

En el estándar MPI2 finalmente se formalizó el mecanismo de creación de procesos, que debe proveer un comando *mpiexec* que permita la ejecución de procesos y cuya sintaxis corresponde a una versión de línea de comandos de la nueva primitiva MPI\_COMM\_SPAWN del estándar [4]:

```
mpiexec -n <numprocs> <program>
```

Las características de los mecanismos de creación de procesos propuestos pueden identificarse en el mecanismo provisto por MPICH.NT, implementación de MPI del Argonne National Laboratory para la plataforma Windows NT [5].

MPICH distribuye utilitarios desarrollados para separar el modo de iniciar un programa MPI del programa en sí mismo, para la gestión centralizada de los recursos de procesamiento disponibles en la red y para garantizar los controles de seguridad.

### **3. Descripción del problema**

Los mecanismos de distribución desarrollados para implementaciones de MPI que ejecutan sobre redes de área local, tienen como característica común ser servicios que brinda el entorno de ejecución y no parte de la biblioteca propiamente dicha.

Los entornos de ejecución cuentan con varias características que limitan la portabilidad de la biblioteca desde el ambiente de una LAN a Internet. Estas características se centran especialmente en lo que refiere a protocolos de comunicación y mecanismos de seguridad, y se mencionan a continuación:

- Utilización de protocolos propietarios sobre TCP-IP para comunicación entre nodos, trabajando con puertos que difícilmente van a poder ser libremente usados en Internet como consecuencia de las restricciones de seguridad.

- Empleo de mecanismos basados en el broadcast para la localización de nodos disponibles, adecuado para redes LAN, pero no aptos para utilizar en Internet.

- Referencia de los hosts a través de sistemas de nombres como el de NetBIOS, no utilizados en Internet.

- Utilización de servicios propietarios de los sistemas operativos para compartir archivos a través de la red como forma de distribución de los ejecutables, cuando estos no están disponibles en cada nodo.

- Manejo de la seguridad a través de los mecanismos de los propios sistemas operativos, tales como dominios administrativos, cuentas de usuario, etc., solución no viable en Internet.

Para construir una biblioteca con características adecuadas para su uso en Internet, el mecanismo de distribución y ejecución remota de procesos diseñado en el marco del proyecto MPI.NET propone resolver estos inconvenientes utilizando herramientas de la tecnología Microsoft .NET, del modo que se describe en la siguiente sección.

### **4. Solución propuesta**

La solución propuesta para resolver el problema fue utilizar el entorno Microsoft .NET para desarrollar un sistema que provea un mecanismo de distribución y un entorno de ejecución con una arquitectura y protocolos adecuados para su uso en Internet, salvando las dificultades planteadas en la sección precedente.

Microsoft .NET es un entorno de desarrollo y ejecución fuertemente orientado a la programación en Internet que brinda facilidades para resolver los problemas que surgen al momento de utilizar la red global como un recurso para el procesamiento distribuido.

Diseñando el sistema sobre los servicios de alto nivel que provee el framework Microsoft .NET, adecuados para la programación en Internet, se construye un sistema potencialmente portable. El framework oculta los servicios del sistema operativo en el que corre, y cualquiera de los componentes de la aplicación es capaz de ejecutar en cualquier plataforma que tuviera su implementación del framework

Las principales facilidades o servicios del entorno utilizados fueron las siguientes:

- **Remoting**  
Remoting es la infraestructura del entorno Microsoft .NET para distribuir y comunicar objetos en escenarios remotos. A través de este mecanismo, Microsoft .NET ofrece un modelo de programación simple que hace la interacción transparente al desarrollador  
Una descripción detallada de los mecanismos de remoting puede encontrarse en el texto de Rammer [6] o en los artículos en línea [7].
- **Reflection**  
El acceso a los metadatos de los assemblies .net brinda la posibilidad de crear instancias de objetos e invocar sus métodos interrogando a los propios assemblies sobre las clases que contienen y sus miembros en tiempo de ejecución
- **Seguridad**  
Los mecanismos de seguridad basada en la evidencia y la autenticación mediante el uso de certificados de seguridad que provee el propio entorno permiten resolver los requerimientos relativos a la misma en el sistema de distribución y ejecución propuesto.

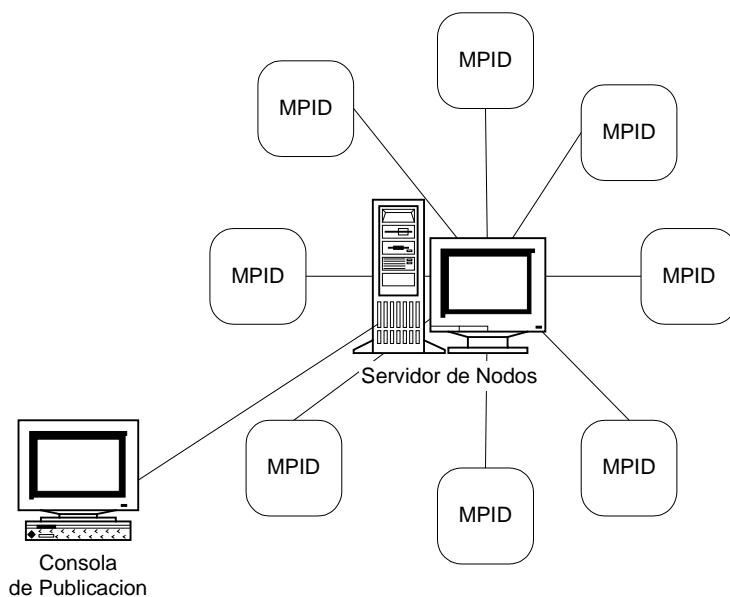
#### 4.1 Arquitectura de la solución

Conjuntamente con el objetivo de utilizar una arquitectura y protocolos adecuados para Internet, fueron considerados como base para el diseño los requerimientos funcionales definidos para un mecanismo de este tipo, de acuerdo al estándar [4]:

- separación del modo de iniciar un programa MPI del programa en sí mismo,
- gestión centralizada de los recursos de procesamiento disponibles en la red y
- ofrecer mecanismos para control de la seguridad.

El sistema se compone de tres aplicaciones que interactúan a lo largo del proceso de ejecución: la Consola de Publicación, el Servidor de Nodos y el demonio (MPID), que resuelven la carga y publicación de un assembly (bloque de construcción de un programa en el framework Microsoft .NET), la distribución de un assembly publicado a un equipo remoto y la ejecución de un assembly distribuido en un equipo remoto, respectivamente.

Un diagrama conceptual de la solución se ofrece en la figura 1



*Figura 1 : Diagrama conceptual de la solución propuesta*

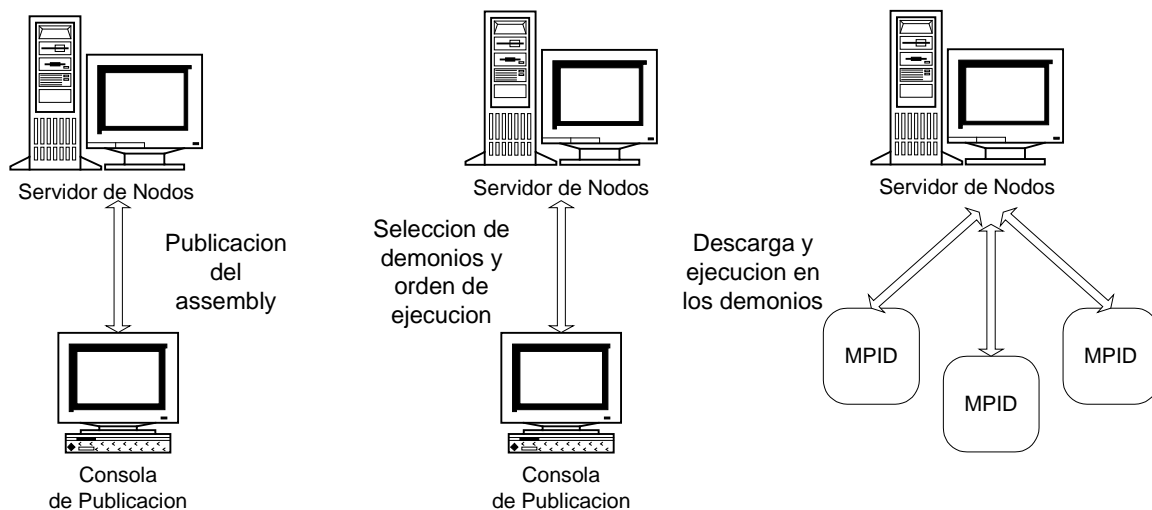
Aplicación	Función
Consola de Publicación	carga y publicación de un assembly
Servidor de Nodos	distribución de un assembly publicado a un equipo remoto
demonio (MPID)	ejecución del proceso en el equipo remoto

*Tabla 1 : Aplicaciones del mecanismo de distribución y ejecución*

El único componente que tiene una interacción con el usuario del sistema es la Consola de Publicación. A través de la Consola se realiza la conexión con el Servidor de Nodos para la publicación de los assemblies a distribuir. Una vez que el usuario tiene un assembly en el Servidor de Nodos, puede seleccionar en la Consola en cuales de los nodos disponibles va a ejecutarlo.

La Consola de Publicación resuelve la distribución del código a ejecutar en los nodos del cluster y su ejecución, usando los servicios que ofrece el Servidor de Nodos a través de sus objetos expuestos. El proceso de distribución y ejecución se resume en la figura 2.

### Proceso de distribución / ejecución



*Figura 2 : Proceso de distribución y ejecución*

## 4.2 Diseño de interfaces y clases

El diagrama de diseño de clases del sistema se presenta en la figura 3.

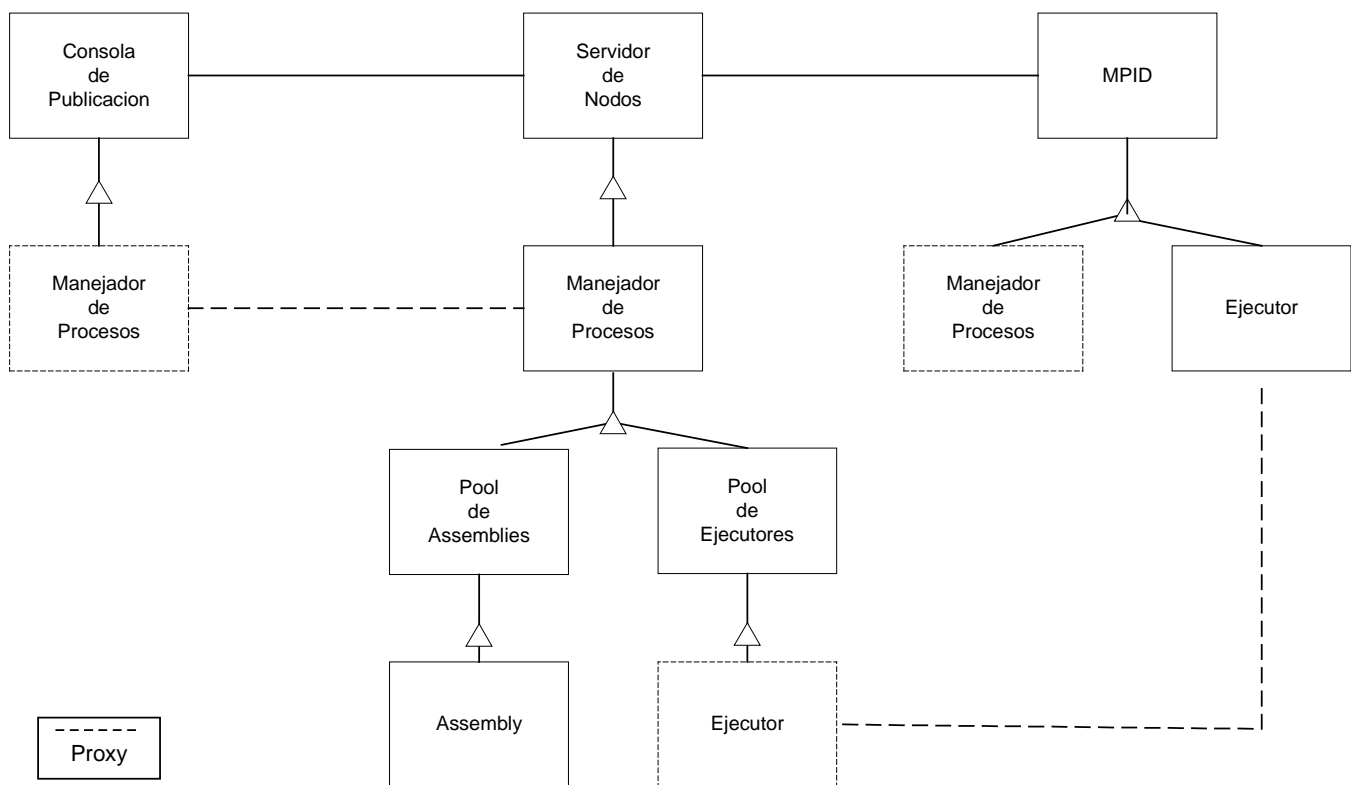


Figura 3 : Diagrama de diseño de clases del sistema

Se creó un namespace *MPIShared* que contiene las dos clases compartidas por las aplicaciones del sistema: *ManejadorDeProcesos* y *Ejecutor*.

La clase *ManejadorDeProcesos* brinda funciones y métodos para manejar una colección de *Assemblies*, tal cual se muestra en la figura a continuación.

MPIShared:: <b>ManejadorDeProcesos</b>
- PoolDeAssemblies : Collection - PoolDeClases : Collection - PoolDeEjecutores : Collection - ds : DataSet
+getPoolAssemblies() : Collection +getPoolClases() : Collection +getPoolEjecutores() : Collection +getDs() : DataSet +CargarProceso(in AssProceso : Assembly, in strclase : String, in strid : String) : Integer +getAssembly(in idassembly : String) : Assembly +getClase(in idassembly : String) : String +registrarDemonio(in shost : String, in sip : String) +referenciarEjecutor(in sipserver : String, in sip : String, in shost : String) +getEjecutor(in sipdemonio : String) : Ejecutor

Figura 4 : Descripción de la clase *ManejadorDeProcesos*

Los métodos de la clase permiten cargar una nueva assembly en la colección con un identificador, descargar determinada *Assembly* de la colección a través de su identificador, etc.

El namespace *MPIShared* también contiene una colección de un objeto *Ejecutor* expuesto por los Demonios a través del cual se ejecutan las clases en ellos.

MPIShared::Ejecutor
-serverip : String -shostname : String -sip : String -m : ManejadorDeProcesos
+setManejador(in manej : ManejadorDeProcesos) +setIpDemonio(in ip : String) +setHostDemonio(in host : String) +getHostDemonio() : String +correrProceso(in idassembly : String)

Figura 5 : Descripción del objeto Ejecutor

El *Manejador de Procesos* contiene un DataSet en el cual se mantiene la información referente al registro de los Demonios activos (conteniendo la dirección IP y nombre de HOST). Complementariamente, provee de las funciones de registro en el DataSet, que son utilizadas por los demonios para registrarse.

Al conectarse con el servidor de nodos, cada demonio se registra utilizando el método *RegistrarDemonio*.

Al activarse, cada demonio crea un objeto de la clase *Ejecutor*. Mediante el método *ReferenciarEjecutor* carga una referencia al objeto creado en la colección de Ejecutores del Manejador de Procesos del Servidor de Nodos.

Las restantes clases del diagrama de la figura 3 corresponden a las aplicaciones del Sistema.

La clase *ServidordeNodos* modela la aplicación del mismo nombre, que debe ser la primera en ejecutarse para garantizar el funcionamiento del sistema.

Cuando se carga la aplicación se crea una instancia de la clase compartida *ManejadorDeProcesos* y se expone mediante Remoting, para que este objeto pueda ser referenciado o accedido por las aplicaciones del Sistema.

Los *Demonios* son las aplicaciones ejecutoras de los procesos y en el marco del proyecto MPI.NET serán quienes proveerán las funcionalidades de comunicación entre procesos distribuidos.

Cada vez que se levanta un *demonio*, éste solicita la dirección IP del Servidor de Nodos para poder obtener la referencia al objeto *ManejadordeProcesos* expuesto por el Servidor. Posteriormente, se crea la instancia del objeto Ejecutor que se expone mediante Remoting para ser referenciado y agregado a la colección en el Servidor de Nodos, con la IP del Demonio como identificador.

El Ejecutor será utilizado por la Consola de Ejecución para ejecutar alguna de las clases publicadas. Por último, se registra el demonio en el Servidor de Nodos como se detallo anteriormente, indicando que esta listo y disponible para ejecutar alguna clase.

Es de destacar que solamente es posible correr un Demonio por máquina ya que no se puede dar más de un uso al par IP/puerto que se usa para exponer al objeto Ejecutor creado por el Demonio.

MPID::MPID
-manejador : ManejadorDeProcesos -ejec : Ejecutor -sipaddress : String -shostname : String
-CONECTAR(in sender : Object, in e : EventArgs) -LEERDISCO(in sender : Object, in e : EventArgs) -DARACCESO(in sender : Object, in e : EventArgs) +QUITARACCESO()

Figura 6 : Descripción de la clase demonio

La Consola de Publicación constituye la interfase con el usuario del sistema. Ofrece información sobre la configuración de la maquina virtual y permite la publicación y ejecución de clases de usuarios en los nodos seleccionados.

Cuando se carga la Consola de Ejecución se solicita la dirección IP del Servidor de Nodos para obtener la referencia al objeto *ManejadordeProcesos* que este expone. Este *ManejadordeProcesos* es el mismo objeto que fue referenciado por todos los *Demonios* y a su vez contiene la colección de objetos referenciados a los objetos Ejecutores expuestos por cada *Demonio*.

ConsolaDePublicacion
-manejador : ManejadorDeProcesos
-sipaddress : String
-shostname : String
-SALIR(in sender : Object, in e : EventArgs)
-CONECTAR(in sender : Object, in e : EventArgs)
-ACTUALIZAR(in sender : Object, in e : EventArgs)
-EJECUTAR(in sender : Object, in e : EventArgs)
-EXAMINAR(in sender : Object, in e : EventArgs)

Figura 7 : Descripción de la clase *ConsoladePublicación*

La Consola muestra una lista con los Demonios registrados, que se inicializa con los datos almacenados en el DataSet contenido en el Servidor, como se puede apreciar en la figura 8.

Desde la Consola es posible publicar assemblies en la colección contenida en el objeto *ManejadordeProcesos* expuesto. Para ello se solicita un identificador para la clase, que se usará para descargar el assembly y ejecutarlo, el lugar físico donde se encuentra la dll y el nombre de la clase. Cada assembly es cargado en la colección a través de funciones expuestas por la clase compartida.

Cabe destacar que cada clase desarrollada por un usuario que pretenda utilizar el sistema debe tener referencia a la biblioteca compartida MPIShared.dll y definir en su especificación un método con nombre *MPI\_RUN ()* que será el punto de entrada o ejecución de la clase.

Estos son los únicos requisitos que deberá cumplir una aplicación para la utilización del sistema de distribución y ejecución, y en general para utilizar todas las potencialidades de la biblioteca MPI.NET que se diseña, más allá de una recompilación en el nuevo entorno de trabajo para programas diseñados en entornos anteriores.

Para ejecutar una clase en el cluster debe referenciarse mediante el identificador con el cual fue publicada. La Consola tiene referencia al objeto *ManejadorDeProcesos* expuesto por el Servidor de Nodos, el cual a su vez contiene la colección de objetos Ejecutores que referencian a los expuestos por cada *Demonio*. La Consola indica al manejador en cuales ejecutores se quiere ejecutar el assembly. El *manejadorDeProcesos* invoca un método que descarga el Assembly de la colección en cada uno de estos ejecutores, creando a través de los mecanismos que brinda reflection una instancia de la clase contenida en esa Assembly y ejecutando su método *MPI\_RUN()*.

Esto mecanismo posibilita la ejecución del assembly en cada uno de los Demonios seleccionados, ya que los objetos Ejecutores fueron creados en los Demonios y expuestos mediante Remoting.



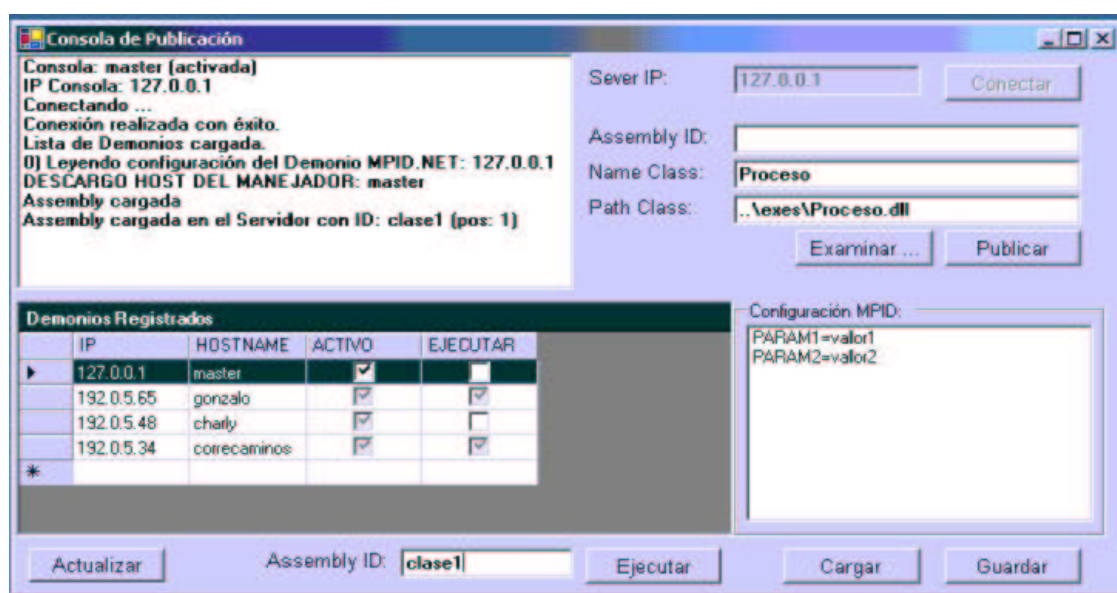


Figura 8 : La aplicación Consola de Publicación

### 4.3 Seguridad

Las cuestiones de seguridad se convierten en un asunto muy importante en un sistema de ejecución distribuida que propone compartir recursos computacionales a través de Internet.

El entorno Microsoft .NET provee un sistema de seguridad amplio, capaz de confinar los programas a ejecutar distribuidamente en contextos de seguridad acotados y definidos de antemano.

Los aspectos principales de las características de seguridad en el entorno Microsoft .NET se analizan detalladamente en el trabajo [8].

El objetivo planteado para el sistema de distribución y ejecución en lo concerniente al control de la seguridad consistió en acotar al proceso que es descargado y ejecutado en una máquina remota a un contexto de seguridad definido desde el propio sistema.

Para ello se definieron políticas de seguridad en el propio proceso demonio, que podrían ser modificables por el usuario que tiene instalado el demonio, de manera de otorgar o quitar permisos sobre la propia máquina del usuario al demonio. Estas mismas políticas de seguridad se aplicarán al proceso que es descargado, ya que su ejecución será invocada dentro del demonio.

En el prototipo del mecanismo de distribución y ejecución diseñado, se definieron únicamente políticas de acceso al filesystem local, pero se aplicarán restricciones más específicas en la versión final del sistema.

En el entorno Microsoft.NET solamente son asignados permisos de seguridad si es que son explícitamente solicitados. Como ejemplo, si solo se solicita la autorización **FileIOPermission** que permite el acceso a archivos, todo otro permiso será negado al no explicitarse su solicitud. Esta regla se cumple aún si hubieran sido adjudicados permisos por defecto de la política de seguridad local, que deja de tomarse en cuenta.

En el prototipo diseñado se utilizaron las herramientas del entorno que se describen en la tabla 2

	Descripción
Imports System.Security	Provee la estructura del Sistema de Seguridad del CLR (Common Language Runtime), incluyendo las clases base para los permisos.
Imports System.Security.Principal	Namespace que define el objeto principal que representa el contexto de seguridad bajo el cual el código corre.
Imports System.Security.Permissions	Namespace que define clases que controlan el acceso a operaciones y recursos basados en políticas.

Tabla 2 : Descripción de mecanismos de seguridad.

Como ejemplo de aplicación de estos mecanismos, para asignar permisos de accesos sobre archivos del disco local, se utiliza:

```
Dim UserPermission As New  
Security.Permissions.FileIOPermission(FileIOPermissionAccess.AllAccess, "c:\")
```

Mientras que para denegar los permisos, debe utilizarse:

```
UserPermission.Deny()
```

En el prototipo se capturan las excepciones de seguridad generadas al momento de intentar acceder a un recurso sin disponer los permisos necesarios, para no abortar la ejecución del programa. Para ellos se utiliza

```
Catch miexcep As System.Security.SecurityException
```

Reportándose un mensaje referente al intento de acceso no autorizado.

## 5. Conclusiones y Trabajo Futuro

Luego de diseñado e implementado, el prototipo fue testeado sobre una máquina virtual distribuida en Internet. Se comprobó su correcto funcionamiento para la distribución y ejecución de códigos de pequeño y mediano tamaño.

En el marco del proyecto MPI.NET, el éxito en el diseño e implementación del sistema de distribución y ejecución de código permite alcanzar uno de los objetivos específicos propuestos. El mecanismo cumple con la totalidad de sus requerimientos y proporciona una solución robusta para el problema planteado, cumpliendo con las especificaciones del estándar MPI.

Por su parte, la utilización de nuevas herramientas de la tecnología ha permitido al grupo de trabajo comprender ciertos valiosos mecanismos que ofrece el entorno Microsoft .NET para programación paralela y distribuida a través de Internet.

Si bien el problema de la distribución de código queda totalmente resuelto con el diseño explicado, la versión final del sistema de distribución deberá incluir ciertas funcionalidades que el prototipo no posee. La configuración dinámica de nodos de la máquina virtual desde la consola es un ejemplo de funcionalidad a implementar en el futuro.

Por otra parte, los aspectos de seguridad se redujeron al mínimo en el prototipo diseñado, ya que se concentró el trabajo en posibilitar la distribución y la ejecución de código. Es claro que trabajando sobre Internet, la seguridad deja de ser un accesorio secundario y pasa a tener una importancia vital para un mecanismo que permita ejecutar códigos “extraños” en equipos remotos. La definición de nuevas políticas de seguridad, incorporando mecanismos de autenticación a través del uso de certificados digitales es una prioridad actual del desarrollo. Los certificados digitales serán utilizados para conectar la consola y cada uno de los demonios al servidor de nodos.

## Referencias bibliográficas

- [1] W. Gropp, E. Lusk, A. Skjellum. *Using MPI, Portable Parallel Programming with the Message Passing Interface*, The MIT Press, ISBN 0-262-57132-3, 1999.
- [2] J. Richter, *Applied Microsoft .NET Framework Programming*, Microsoft Press, ISBN 0735614229, 2002.
- [3] Estándar MPI1, MPIForum, <http://www.mpi-forum.org/docs/mpi-10.ps.Z> , consultado junio 2002
- [4] Estándar MPI2, MPIForum, <http://www.mpi-forum.org/docs/mpi-20.ps.Z> , consultado junio 2002
- [5] MPICH.NT Documentation, Argonne National Laboratories <http://www-unix.mcs.anl.gov/mpi/mpich/> , consultado junio 2002.
- [6] I. Rammer, *Advanced .NET Remoting*, APRESS; ISBN 1590590252, abril 2002.
- [7] D. Talbot, *Introducing .NET Remoting* <http://www.vbweb.co.uk/show/1916>, consultado junio 2002.
- [8] D. Watkins, *An Overview of Security in the .NET Framework*, White Paper, Microsoft Corporation, 2002.