Manual de uso de las interfases

ConnectionManager

Esta interfaz es el punto de entrada de un nodo al cluster. Es la única interfaz que debe ser publicada vía RMI (si bien las demás interfases también son usadas remotamente, solo a esta debe hacerse un bind explícito). Al registrar esta interfaz, se debe utilizar el puerto default de RMI, caso contrario, una implementación de un alumno puede no llegar a conectarse a un nodo otro alumno. Además de utilizar el mismo puerto, también se debe utilizar el mismo nombre lógico. Dicho nombre lógico se encuentra en la interfaz ReferenceName.
La interfaz ConnectionManager debe utilizarse para poder acceder a todas las demás interfases de un nodo. Debe ser el punto de entrada, tanto para los requerimientos remotos como locales.

ClusterAdministration

En esta interfaz se encuentran todos los métodos necesarios para la administración de un cluster. Es mediante la misma que se crea el cluster o grupo de trabajo, se agregan y se eliminan nodos del cluster. Es la primer interfaz a ser usada en la aplicación, debido a que antes de realizar cualquier simulación es necesario crear un cluster.

ClusterCommunication

Esta interfaz es utilizada por los nodos del cluster para poder comunicarse entre sí. El primer método sirve para mandar mensajes a todos los nodos, mientras que el segundo método sirve para mandar un mensaje entre dos nodos. Esta interfaz es utilizada constantemente por el cluster para realizar cualquier tipo de comunicación. Por ejemplo, para realizar el balanceo de carga, migración de agentes, migración de recursos, informar la existencia de un nuevo nodo, informar la baja de un nodo, etc. No hace falta aclarar que si esta interfaz no funciona correctamente, nada de lo anterior va a poder funcionar.

MessageListener

Cada nodo, además de tener la habilidad de enviar mensajes, debe poder recibirlos. Para esto, se debe utilizar esta interfaz en un thread separado (para que no afecte al funcionamiento del resto del nodo). Esta interfaz posee el método a invocar ante la llegada de un mensaje nuevo, o en caso de recibir un pedido de mensajes desde otro nodo.

Payload

Debido a que todas las implementaciones realizadas por los alumnos deben poder comunicarse entre sí, es necesario fijar un formato de mensajes enviados. Todos los mensajes tienen, además de información del nodo que generó el mensaje, datos específicos. Por ejemplo, un mensaje utilizado para informar la baja de un nodo deberá tener como dato el nodo desconectado, pero un mensaje de migración de agentes deberá tener información sobre el agente migrado. Para solucionar esto, se diseño la interfaz Payload, con los métodos necesarios en cada uno de los casos. Esta interfaz es utilizada solamente para el envío y recepción de mensajes.

Transactionable

Cuando dos nodos deben realizar una transacción de recursos (un nodo envía cierta cantidad de cierto recurso hacia otro nodo), se debe asegurar que todo el sistema quede en un estado concistente. Es decir, que un error de un nodo no signifique que se pierdan recursos o que se dupliquen. Entonces, ante este evento y utilizando esta interfaz se debe crear una transacción entre dos nodos. El flujo normal es el siguiente:

1. Un nodo intenta crear una transacción utilizando el método “beginTransaction”
2. Dicho nodo debe asegurarse que el nodo destino puede crear la transacción, es decir que no se encuentra en otra. Esto es realizado mediante el método “acceptTransaction”
3. Durante la transacción, se realiza el intercambio de recursos utilizando el método “exchange”.
4. Una vez realizado el intercambio, se finaliza la transacción con el método “endTransaction”

Este último paso es en el cual se persisten los cambios en ambos nodos. Para esto, se utiliza la siguiente interfaz.

ThreePhaseCommit

Interfaz con los métodos necesarios para poder realizar un commit distribuido en 3 pasos. Posee todos los métodos necesarios para iniciar el protocolo, finalizar el protocolo, abortar la transacción por error, abortar la transacción por timeout, etc.
Esta interfaz debe ser utilizada solamente dentro de un contexto de transacción creado por la interfaz anterior.