DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 3

Sistemas Distribuidos

23 de Junio de 2016 — Sistemas Operativos

Integrante	LU	Correo electrónico
Costa, Manuel José Joaquin	035/14	manucos94@gmail.com
Coy, Camila Paula	033/14	camicoy94@gmail.com
Ginsberg, Mario Ezequiel	145/14	ezequielginsberg@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Introducción

En este trabajo práctico se nos pide desarrollar un algoritmo de elección de líder en un anillo de procesos de un sistema distribuido. Para ello vamos a usar de una implementación de MPI, las funciones no bloqueantes que provee la misma.

2. Consideraciones

Durante el desarrollo del trabajo descubrimos que cuando todos los procesos ponen a circular una elección de líder, el programa termina con error o sin un líder definido debido a que todos los procesos envían un mensaje al siguiente y esperan el ACK de forma bloqueante, lo que ocasiona que nunca puedan recibir la tupla de comienzo de nueva elección del proceso anterior para enviarle su ACK al anterior. Por ejemplo, si tenemos 3 procesos (1, 2 y 3, respectivamente) y los tres inician una elección, puede ocurrir la siguiente secuencia:

- El proceso 1 le envía la tupla <1,1>al proceso 2 y se queda esperando que el proceso 2 le responda "OK"
- El proceso 2 le envía la tupla <2,2>al proceso 3 y se queda esperando que el proceso 3 le responda "OK"
- El proceso 3 le envía la tupla <3,3>al proceso 1 y se queda esperando que el proceso 1 le responda "OK"
- El proceso 3 finaliza su timeout y le va a enviar al proceso 2 la tupla <3,3>
- El proceso 1 finaliza su timeout y le va a enviar al proceso 3 la tupla <1,1>
- El proceso 2 finaliza su timeout y le va a enviar al proceso 4 la tupla <2,2>, pero el proceso 4 no existe y ahí el programa termina con un error.

Si no ocurre lo anterior es porque el timeout del programa finalizó antes de que el proceso 2 envíe un mensaje a 4 lo que ocasiona que todos los procesos impriman por pantalla que no son el líder.

Una forma de resolver este problema es agregar al inicio de la función *iniciar_eleccion* un *MPI_Iprobe* y si recibimos un mensaje, responderlo con "OK". Con esto lo que logramos es responder el ACK al proceso que nos envió el mensaje y evitar el deadlock (espera circular).

Otra forma de resolver este problema puede ser modificando la función *main* para evitar que todos los procesos invoquen a la función *iniciar_eleccion*.

Otro caso en el que el programa puede fallar es cuando el último proceso no responde el ACK por diversos motivos (que esté esperando una respuesta de otro proceso, que esté enviando el ACK a otro, etc) y como nosotros suponemos que el último siempre responde, cuando se acaba el timeout de espera del ACK, el proceso le envía el mensaje al pid = último+1, lo cual provoca un error del tipo "invalid rank".

3. Desarrollo

A continuación detallaremos las dos funciones que se nos pidió implementar.

3.1. Inicio de Elección

Para iniciar la elección lo que hacemos es enviar una tupla de ints <MI_ID,MI_ID>al siguiente proceso y esperamos que éste responda con un "OK" que recibió el mensaje. Si pasa un segundo y el proceso todavía no recibió la respuesta, entonces le envía el mensaje al siguiente del siguiente en el anillo, suponiendo que éste no era el último, ya que siempre esta vivo. La anterior secuencia de acciones se realizará de forma cíclica mientras que no le respondan.

3.2. Elección del Líder

Al empezar el algoritmo, el proceso espera a que le llegue un mensaje. Cuando le llega le responde al emisor con un ACK y pasa a mirar el contenido del mensaje. Dependiendo del contenido puede ocurrir una de las siguientes situaciones:

- Si el que inició la elección es el proceso que recibió el mensaje, entoces hay 2 posibles acciones:
  - El candidato a líder está más adelante, en cuyo caso se envía un mensaje al siguiente con la tupla <cl,cl>para así avisar al candidato a líder que es el nuevo líder.
  - El proceso es el líder, por lo que actualiza su status a líder.
- Todavía no terminó la primera vuelta, entonces se fija quíen es el candidato a líder:
  - Si el proceso es el nuevo candidato a lider éste envía al siguiente la tupla <inicio,MI_ID>.
  - Sino, envía al siguiente la misma tupla que recibió para que continue la elección.

En todos los casos el proceso espera que le llegue el ACK y de la misma forma que en *iniciar_eleccion* si no lo recibe en un segundo envía al siguiente del siguiente, hasta que le respondan.

Además, si en algún momento de la ejecución se alcanza el timeout asignado (10 segundos por defecto) a un proceso, el mismo informa el resultado parcial obtenido (líder o no líder) y finaliza su ejecución.

4. Conclusiones

Durante el desarrollo del corriente trabajo práctico pudimos experimentar con algoritmos sobre sistemas distribuidos que nos ayudaron a entender un poco más cómo se sincronizan y se comunican las distintas partes del sistema.