Trabajo práctico - Algoritmos en sistemas distribuidos

Sistemas Operativos - 1er Cuatrimestre 2016

Fecha de entrega: Miércoles 22 de junio de 2016 a las 23:00hs

1. Introducción

En los sistemas operativos distribuidos o de red muchas veces existen varios procesos, distribuidos en distintos puntos de la red, con capacidad para responder a ciertas consultas. Si bien esto brinda redundancia en caso de que alguno de los procesos se caiga, puede ser problemático cuando cada consulta debe ser respondida una sola vez. Se trata de casos en los que el cliente podría "confundirse" si recibe múltiples respuestas.

Para solucionar este problema, los distintos procesos corren un algoritmo conocido como *elección de líder*, que permite elegir a uno de los procesos vivos de manera tal que todos estén de acuerdo en que ése es el elegido.

Este TP consiste en implementar una versión simplificada del algoritmo de elección de líder.

2. Escenario

Tendremos una serie de procesos que conformarán un anillo lógico basado en su numeración. El sistema operará en dos fases.

En la primera etapa se realizarán altas y bajas de procesos. Un evento externo, controlable por nosotros, marcará el fin de la etapa.

En la segunda, que durará una cantidad predeterminada de tiempo, realizarán una elección de líder. En esta etapa no morirán ni se agregarán procesos.

Cada sucesión de estas dos etapas conforma una ronda, en la cual podrían iniciase varias elecciones de líder en simultáneo. Al final de la ronda todos los procesos deben imprimir si son o no líderes, y sólo uno de ellos debe ser el líder.

Se elegirá como líder al que tenga la numeración mayor. La dificultad radica en que los procesos no conocen el tamaño del anillo ni el número del mayor.

3. Algoritmo

- Los procesos conforman un anillo lógico, de tamaño conocido sólo por el último.
- Si bien al comienzo la numeración es continua, al morir los procesos podrían quedar huecos.
- Los procesos que mueren no notifican a nadie de su desaparición.
- Los procesos que se añaden, lo hacen al final del anillo. Para hacer eso notifican al que antes era el último proceso. Esta parte del protocolo es provista por la cátedra.
- Cada proceso tiene:
    - Una variable entera `ID`, con su identificador.

    - Una variable booleana que indica si es el último del anillo.
    - Una variable de `status` que comienza valiendo `no líder`.
    - Una variable entera `siguiente` con el id del siguiente proceso del anillo, que comienza valiendo `ID` + 1.
- Al comienzo de la primera etapa cada proceso decide al azar si comienza una elección de líder o no.
- Para comenzarla, debe circular por la red un par de enteros <i, cl>, donde i es el id del iniciador de la elección, y cl el del candidato a líder hasta el momento. Este mensaje es enviado al siguiente proceso dentro de anillo con los valores `<ID, ID>`.
- Cuando se debe enviar un mensaje al siguiente proceso, se lo envía a aquél cuyo id figura en la variable `siguiente`. Si pasados t segundos no recibió un ACK, intentará con el proceso de id `siguiente+1`, actualizando el valor de `siguiente`. El último proceso del anillo, que sabrá que es el último, considerará al proceso número 1 como su siguiente.
- Cuando se recibe un mensaje de elección de líder <i, cl>, se actúa de la siguiente manera:
    - Hay una elección de líder dando vueltas. `status`:= `no líder`.
    - Si i==ID, la elección dio toda la vuelta al anillo, y cl es el líder. Si cl>ID, significa que el líder está más adelante y no sabe que ganó. El token debe seguir girando con los valores <cl, cl>. Si cl==ID, soy el líder y debo actualizar `status` a `líder`.
    - Si i!=ID, el token debe seguir girando. Si ID>cl, hay un nuevo candidato a líder. Es decir, cl:= ID.

4. Implementación

Para implementar el TP utilizaremos MPI[1] y haremos algunas simplificaciones.

El proceso 0 será un proceso de control que se encargará de poner a correr y matar al resto de los procesos del anillo. En realidad, todos los procesos estarán corriendo desde el comienzo, y el proceso de control hará simplemente una activación lógica de los mismos.

Dicho proceso de control ya se brinda implementado por la cátedra, como así también el grueso del código necesario para el TP. Cada uno de los procesos que componen el anillo tiene implementada ya la funcionalidad para unirse y abandonar al anillo, incluyendo la lógica para saber su ID y si es o no el último del anillo.

El protocolo de elección de líder debe programarse en las funciones `iniciar_eleccion()` y `eleccion_lider()` en el archivo `eleccion.c`.

Dichas funciones deben programarse de manera prolija y documentarse apropiadamente.

Para hacerlo, utilizaremos las siguientes funciones de MPI:

- `MPI_Isend()` Envío no bloqueante de mensajes.
- `MPI_Iprobe()` Indica si hay o no mensajes, de manera no bloqueante.
- `MPI_Irecv()` Lee un mensaje de manera no bloqueante.

Una buena descripción de ellas se encuentra en `https://computing.llnl.gov/tutorials/mpi` .

Notar que se provee un makefile para GNU Make (`GNUmakefile`) donde puede ser necesario modificar la variable `MPIROOT`.

[1] `http://www.mpi-forum.org/docs/docs.html`

5. Ejecución

Para ejecutar el programa compilado debe usarse:

```
mpirun <--no-daemonize> <-all-local> -np N
```

donde `N` es la cantidad de procesos a lanzar y las opciones entre llaves podrían ser necesarias o no dependiendo de la implementación de MPI que se use.