Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

TP3 - Sistemas Distribuidos

Sistemas Operativos

Primer Cuatrimestre de 2016

Apellido y Nombre	LU	E-mail
Casanova, Manuel	355/05	morbous_panda@hotmail.com
Tripodi, Guido	843/10	guido.tripodi@hotmail.com

Contents

1 Parte I – Algoritmo

1.1 Algoritmo

- **Implementacion** Nuestro algoritmo se implemento de la siguiente manera:

  Chequeamos si tenemos un mensaje, en caso de tenerlo revisamos el tipo de TAG, para verificar si el mismo es un ACK o un token de otro proceso que se encuentra en circulación en el anillo.
  Una vez realizado dicha verificación, si el mismo es un **TAG_OTORGADO** (utilizamos dicho tag para enviar y recibir token) procedemos a chequear los valores token[0] y token[1] como se solicito.
  Una vez verificado si el token recibido es propio o ajeno y si soy lider o no procedemos a actualizarlo con los respectivos datos:

  Cuando se recibe un mensaje de eleccion de lider <i, cl>, se actua de la siguiente manera:
  • Hay una eleccion de lıder dando vueltas. status:= no lıder.
  • Si i==ID, la eleccion dio toda la vuelta al anillo, y cl es el lıder. Si cl>ID, significa que el lıder esta mas adelante y no sabe que gano. El token debe seguir girando con los valores <cl, cl>. Si cl==ID, soy el lıder y debo actualizar status a lıder.
  • Si i!=ID, el token debe seguir girando. Si ID>cl, hay un nuevo candidato a lıder. Es decir, cl:= ID.

  En caso de no ser lider, procedemos a enviar el token al próximo proceso del anillo, aquí quedamos en espera de un ACK del próximo pid, si pasado X tiempo no recibimos dicho ACK le enviamos el mismo token al próximo, así hasta que alguno de dichos procesos nos responda o peguemos la vuelta al anillo. (En la parte 2 de este informe aclararemos ciertos puntos referidos a este tema ya que dicha espera suele perjudicar ampliamente a la hora de que el token viaje por el anillo buscando la elección correcta del líder)

  En cambio, si recibimos un **TAG_ACK**, verificamos si estabamos en espera del mismo en caso afirmativo, dejamos en_espera = 0 y actualizamos el valor de nuestro proximo pid actual.
  Si recibimos un ACK el cual el origen no se encuentre dentro de los rangos pid+1 <= origen <= proximo levantaremos el mismo pero no cambiaremos el valor de espera.

2 Parte II: Puntos importantes

2.1 Puntos importantes

Luego de varios testeos, hemos llegado a la conclusión que al tener varios procesos poniendo a circular una elección, y tener una espera de ack por envio de token a otros procesos, puede generar que varios mensajes se encolen, y si dicha espera aumenta por la falta de respuesta, existe la posibilidad que el proceso ultimo no llegue a desencolar todos los mensajes que recibio y de esta forma no llegar a enterarse que es LIDER. Una de las causas de esto se da ya que al tener varios procesos de forma aleatoria poniendo a circular elecciones dicha espera puede aumentar por tener a varios procesos esperando ACK.

Una de las formas que realizamos para que dicha probabilidad disminuya fue antes de enviar un token y quedar en espera chequear si recibimos un nuevo token para responderle al origen y no generar una espera circular entre varios procesos.

3 Bibliografía

- Cátedra de Sistemas Operativos - Clases teóricas y prácticas (1º Cuatrimestre 2016)