Replicación y consistencia

Tema 3

Comunicación en grupos

Contenidos Tema 3

1.- Objetivos.
2.- Conceptos preliminares.
3.- Pertenencia: Propiedades.
4.- Difusión: Propiedades.
5.- Ordenación de difusiones.
6.- Algoritmos.

Bibliografía

[BSS91] K. P. Birman, A. Schiper, P. Stephenson: “Lightweight Causal and Atomic Group Multicast”, ACM Trans. on Comp. Sys., 9(3):272-314, agosto 1991.

[CKV01] Gregory Chockler, Idit Keidar, Roman Vitenberg: “Group Communication Specifications: A Comprehensive Study”, ACM Computing Surveys, 33(4):427-469, 2001.

[HT93] Vassos Hadzilacos, Sam Toueg: “Fault-Tolerant Broadcasts and Related Problems”. En S. J. Mullender, editor, “Distributed Systems”, capítulo 5, págs. 97-145. Addison-Wesley, 2ª edición, 1993.

Contenidos Tema 3

1.- **Objetivos.**
2.- Conceptos preliminares.
3.- Pertenencia: Propiedades.
4.- Difusión: Propiedades.
5.- Ordenación de difusiones.
6.- Algoritmos.

1. Objetivos

- La comunicación en grupos tiene como objetivos:
  - Proporcionar ciertas garantías en la entrega de los mensajes difundidos a un grupo de procesos:
    - Fiabilidad: Que todos los mensajes se entreguen a todos los procesos activos.
    - Orden: Que se respeten ciertas condiciones en el orden de entrega.
    - Uniformidad.

1. Objetivos

- Aprovechar los servicios proporcionados por los detectores de fallos o monitores de pertenencia.
  - De ellos surge el concepto de “vista”.
  - Se toma esta como base para definir la “sincronía virtual”.
    - Los mensajes difundidos deben ser entregados en todos sus procesos destinatarios en una misma vista.
    - Esto simplifica la implantación de aplicaciones con componentes replicados.

Contenidos Tema 3

1.- Objetivos.

2.- **Conceptos preliminares.**

3.- Pertenencia: Propiedades.

4.- Difusión: Propiedades.

5.- Ordenación de difusiones.

6.- Algoritmos.

2. Conceptos preliminares

- **Grupo**: Un conjunto de procesos que tiene algún objetivo común y que podrá utilizarse como destinatario de ciertas difusiones.
  - Se utilizan principalmente para modelar al conjunto de réplicas de un determinado servicio.
  - Pero también pueden emplearse para modelar cualquier tipo de componente (objeto, proceso) en una aplicación distribuida.

2. Conceptos preliminares

- **Vista**: La salida proporcionada por un servicio de pertenencia.
  - Cada vez que haya un cambio en el conjunto monitorizado, se genera un cambio de vista.
  - Interesa que las entregas de los mensajes difundidos y de las notificaciones de cambio de vista se realicen en el mismo orden en todos los componentes del grupo.

2. Conceptos preliminares

- **Comunicación de grupos orientada a vistas**: Cuando se tiene en cuenta la interacción entre el servicio de pertenencia y los protocolos de difusión.
  - [CKV01] toma esta aproximación.
    - En las secciones 3 y 4 tomaremos ese trabajo como referencia básica.
  - [HT93] no estudia los servicios de pertenencia.

Contenidos Tema 3

1.- Objetivos.

2.- Conceptos preliminares.

3.- **Pertenencia: Propiedades.**

4.- Difusión: Propiedades.

5.- Ordenación de difusiones.

6.- Algoritmos.

3. Pertenencia: Propiedades

- Según [CKV01] un servicio de pertenencia debe satisfacer estas propiedades:
  - Seguridad:
    - Autoinclusión.
    - Orden monótono local.
    - Evento de vista inicial.
  - Viveza:
    - Las propiedades de viveza se definen de manera conjunta para pertenencia y difusión. Se estudiarán en la sección 4.

3.1. Propiedades de seguridad

- **Prop. 1: Auto-inclusión**: Si un proceso P instala una vista V, entonces P es un miembro de V.
  - Los procesos que gestionan la pertenencia deben formar parte del grupo monitorizado.

3.1. Propiedades de seguridad

- **Prop. 2: Orden monótono local**: Si un proceso P instala una vista V tras instalar V', entonces el identificador de V es (estrictamente) mayor que el identificador de V'.
  - No puede ocurrir que dos vistas distintas tengan un mismo identificador.
  - Los identificadores aseguran un orden cronológico dentro de la historia de vistas.
  - Esto no obliga a que todos los procesos tengan que ver la misma secuencia de vistas.

3.1. Propiedades de seguridad

- **Prop. 3: Evento de vista inicial**: Cada evento relacionado con la comunicación (envío, entrega,...) ocurre en alguna vista.
  - Esto permite dos formas de construir tal vista inicial:
    - Antes de difundir/entregar ningún mensaje, el servicio de pertenencia es ejecutado para determinar la vista inicial.
    - Utilizar una vista por omisión (bien sólo el propio proceso o todos los preconfigurados).

3.1. Propiedades de seguridad

- *Modelo de particionado*. Existen dos opciones para gestionar las particiones en la red:
  - *Modelo de componente primaria* (o de partición primaria): Las vistas instaladas por todos los procesos del sistema están totalmente ordenadas.
    - Esto obliga a que los grupos minoritarios no puedan progresar (i.e., instalar vistas).
  - *Modelo particionable*: Las vistas están ordenadas parcialmente, por lo que puede haber distintas vistas concurrentes (una en cada subgrupo).

3.1. Propiedades de seguridad

- En el modelo de componente primaria se debe cumplir:
  - **Prop. 4: Pertenencia de componente primaria**: Para toda vista V (que no sea la inicial) existen una vista V', tal que V sucede a V', y un miembro P de V que ha instalado V en V'.
    - Esto implica que para todo par de vistas consecutivas, existe al menos un proceso que ha sobrevivido desde la primera a la segunda.

Contenidos Tema 3

1.- Objetivos.

2.- Conceptos preliminares.

3.- Pertenencia: Propiedades.

4.- **Difusión: Propiedades.**

5.- Ordenación de difusiones.

6.- Algoritmos.

4. Difusión: Propiedades

- En [CKV01] se proponen las siguientes propiedades para las difusiones:
  - Seguridad:
    - Integridad en la entrega.
    - No duplicidad.
    - Propiedades relacionadas con la sincronía:
      - Entrega en la vista de envío.
      - Entrega en la misma vista.
      - Sincronía virtual.

4. Difusión: Propiedades

- Viveza: Se deben cumplir simultáneamente estas cuatro propiedades:
  - Precisión en la pertenencia.
  - Viveza en la difusión.
  - Autoentrega.
  - Viveza en la indicación de seguridad.

4.1. Propiedades de seguridad

- **Prop. 1: Integridad en el entrega**: Por cada evento de entrega debe haber un evento de envío del mismo mensaje.
  - Se implementa de manera trivial en todo sistema de comunicación de grupos.

4.1. Propiedades de seguridad

- **Prop. 2: No duplicidad**: No pueden darse dos eventos diferentes de entrega con el mismo contenido en un mismo proceso.
  - Es decir, un mismo mensaje no puede entregarse dos veces en un mismo destino.

4.1. Propiedades de seguridad

- **Prop. 3: Entrega en la vista de envío**: Si un proceso P entrega un mensaje m en la vista V, y algún proceso Q envió m en la vista V', entonces V=V'.
  - Isis y Totem implantaron esta propiedad.
  - En Horus se puede configurar el GCS para que la cumpla. Se la llama "*sincronía virtual fuerte*".
  - Para poder soportar esta propiedad existen dos opciones: descartar los mensajes pendientes de entrega al darse un cambio de vista o bloquear el envío antes de instalar una nueva vista.

4.1. Propiedades de seguridad

- **Prop. 4: Entrega en la misma vista**: Si dos procesos P y Q entregan un mensaje m, ambos lo entregan en la misma vista.
  - Esta propiedad es más débil que la anterior, pero elimina la necesidad de bloquear los envíos o descartar mensajes.

4.1. Propiedades de seguridad

- **Prop. 5: Sincronía virtual**: Si dos procesos P y Q instalan la misma vista V sobre la misma vista previa V', entonces cualquier mensaje entregado por P en V' fue también entregado por Q en V'.
  - Tendremos sincronía virtual si se ha cumplido cualquiera de las dos propiedades anteriores.
  - La sincronía virtual asegura la consistencia de las réplicas cuando se utilicen los modelos de partición de componente primaria, el activo de replicación y el de "fallo parada".

4.2. Propiedades de viveza

- Para poder especificar las propiedades de viveza listadas previamente, necesitamos las siguientes definiciones:
  - Red viva.
  - Componente estable.
  - Detectores de fallos eventualmente perfectos. (Clase diamante-P del tema anterior)
  - Mensajes seguros.

4.2. Propiedades de viveza

- **Def. 1: Red viva**: Si existe un punto en una ejecución tras el que dos procesos P y Q están activos y el canal entre ellos también, entonces tras dicho punto todo mensaje enviado por P eventualmente será entregado en Q.
  - Esto se asume como hipótesis en todas las propiedades de viveza.

4.2. Propiedades de viveza

- **Def. 2: Componente estable**: Un componente estable es un conjunto de procesos que están eventualmente activos y conectados entre sí y para los que todos los demás canales que lleguen a ellos (desde otros procesos no incluidos en el componente) no funcionan.
  - En otros trabajos, a cada componente estable se le llamaba “subgrupo” o “partición”.

4.2. Propiedades de viveza

- **Mensaje seguro**: Un mensaje seguro m es entregado a una aplicación en el proceso P sólo cuando el GCS de P sabe que el mensaje es estable.
  - Un mensaje es estable cuando todos los miembros de una determinada vista lo han recibido.
  - En tal caso, cada miembro de la vista lo podrá entregar, excepto si fallara.

4.2. Propiedades de viveza

- En [CKV01] se separa la notificación sobre la seguridad de un mensaje de su entrega.
  - Esto permite entregar de inmediato tales mensajes, sin esperar a su notificación de seguridad.
  - También origina las dos definiciones que se presentan en la próxima hoja.

4.2. Propiedades de viveza

- **Def. 3: Prefijo de notificación segura**: Si un mensaje es notificado como seguro, entonces es estable en la vista en la que fue entregado.
- **Def. 4: Prefijo fiable de notificación segura**: Si un mensaje m es notificado como seguro en algún proceso P y m ha sido también entregado por Q en la vista V, entonces todo mensaje entregado antes que m en V es también estable en V.

4.2. Propiedades de viveza

- **Prop. 1: Pertenencia precisa**: Un servicio de pertenencia es preciso si para todo componente estable S, existe una vista V con el conjunto de miembros S tal que V es la última vista de todo proceso P en S.
  - La pertenencia precisa es una propiedad tan exigente como los detectores de fallos eventualmente perfectos.

4.2. Propiedades de viveza

- **Prop. 2: Viveza en la difusión**: Todo mensaje enviado por P en V es entregado por todo proceso en S.
  - Donde la relación entre V y S es la especificada en la propiedad anterior.
  - En algunos trabajos, esta propiedad se define como “Finalización de la entrega”:
    - Si un proceso P envía un mensaje m en una vista V, entonces para todo miembro Q de V, o Q entrega m o bien P instala una nueva vista V’ tras V.

4.2. Propiedades de viveza

- **Prop. 3: Auto-entrega**: P entrega cualquier mensaje enviado por él en cualquier vista, a menos que falle tras enviarlo.
- **Prop. 4: Viveza de las notificaciones seguras**: Todo mensaje enviado por P en V es notificado como seguro por todos los procesos en S.

Contenidos Tema 3

1.- Objetivos.
2.- Conceptos preliminares.
3.- Pertenencia: Propiedades.
4.- Difusión: Propiedades.
5.- **Ordenación de difusiones.**
6.- Algoritmos.

5. Ordenación de difusiones

- En [CHK01] se distinguen los siguientes tipos de difusiones:
  - FIFO.
  - Causal.
  - De orden total.

5.1. Difusión FIFO

- **Difusión fiable FIFO***:* Si un proceso P difunde un mensaje *m* antes de *m'* en la vista V, entonces ningún proceso Q entrega *m'* a menos que haya entregado previamente *m*.

5.2. Difusión causal

- En algunos casos el orden FIFO no será suficiente, sobre todo si existen múltiples componentes que puedan interactuar.
- La difusión causal utiliza orden causal, para el cual las órdenes de difusión y entrega de mensajes serán vistas como eventos.
- Un evento *e* **precede causalmente** a un evento *f* , denotado como *e* –––> *f*, si:
  - un mismo proceso ejecuta *e* y *f*, en ese mismo orden, o ...
  - *e* es la difusión de un mensaje *m* y *f* es la entrega de *m*, o ...
  - hay un evento *h*, tal que *e* –––> *h* y *h* –––> *f*.

5.2. Difusión causal

- **Difusión fiable causal**:
  - Si la difusión de un mensaje *m* precede causalmente a la difusión de un mensaje *m'*, y ambos fueron difundidos en la misma vista, entonces ningún proceso entrega *m'* a menos que haya entregado previamente *m*.

5.3. Difusión de orden total

- La difusión causal no establece ningún orden entre los mensajes concurrentes.
- Esto puede plantear problemas en algunas aplicaciones.
- La difusión de orden total corrige este problema exigiendo que todos los procesos entreguen todos los mensajes en el mismo orden.
- Para ello, se añade el orden total:
  - *Difusión fiable de orden total:* Existe una función temporal f() tal que si un proceso Q entrega un mensaje m', y los mensajes m y m' fueron difundidos en la misma vista, y f(m) < f(m'), entonces Q entrega m antes que m'.

5. Ordenación de difusiones

- En [HT93] se utilizan otros nombres para varios tipos de difusión:
  - Difusión de orden total --> Difusión atómica.
    - Pero distingue varios tipos de difusión atómica: atómica, atómica FIFO, atómica causal.
  - Difusión segura --> Difusión uniforme.

Contenidos Tema 3

1.- Objetivos.
2.- Conceptos preliminares.
3.- Pertenencia: Propiedades.
4.- Difusión: Propiedades.
5.- Ordenación de difusiones.
6.- **Algoritmos.**

6. Algoritmos

- En [BSS91] se describen dos algoritmos de difusión empleados en ISIS:
  - CBCAST: Difusión causal.
  - ABCAST: Difusión atómica causal (o de orden total causal).
- Para garantizar el orden causal, se emplean **relojes vectoriales**.
- Para tratar los fallos se emplea sincronía virtual con entrega en la vista de envío.
  - Asume un modelo de fallos de parada.

6.1. Relojes vectoriales

- Un reloj vectorial para un proceso p_i, denotado $RV(p_i)$, es un vector de longitud n (n = nº de procesos en el grupo), indexado por el número de proceso:
  - Cuando p_i inicia su ejecución, $RV(p_i)$ se inicializa a ceros.
  - Por cada evento *send(m)* en p_i, $RV(p_i)[i]$ se incrementa en 1.
  - Cada mensaje difundido por p_i es marcado con el valor incrementado de $RV(p_i)$.
  - Cuando un proceso p_j entrega un mensaje m de p_i conteniendo $RV(m)$, p_j modifica su reloj vectorial de la siguiente manera:

$$\forall k \in 1..n : RV(p_j)[k] = max(RV(p_j)[k], RV(m)[k])$$

6.1. Relojes vectoriales

- Reglas para comparar relojes vectoriales:
  - $RV_1 \leq RV_2 \;\; sii \;\; \forall i : RV_1[i] \leq RV_2[i]$
  - $RV_1 < RV_2 \;\; si \;\; RV_1 \leq RV_2 \;\wedge\; \exists i : RV_1[i] < RV_2[i]$

- Se puede demostrar que dados dos mensajes *m* y *m'*, *m –> m'* sii *RV(m) < RV(m').* Por tanto, los relojes vectoriales permitirán representar la relación de precedencia causal.

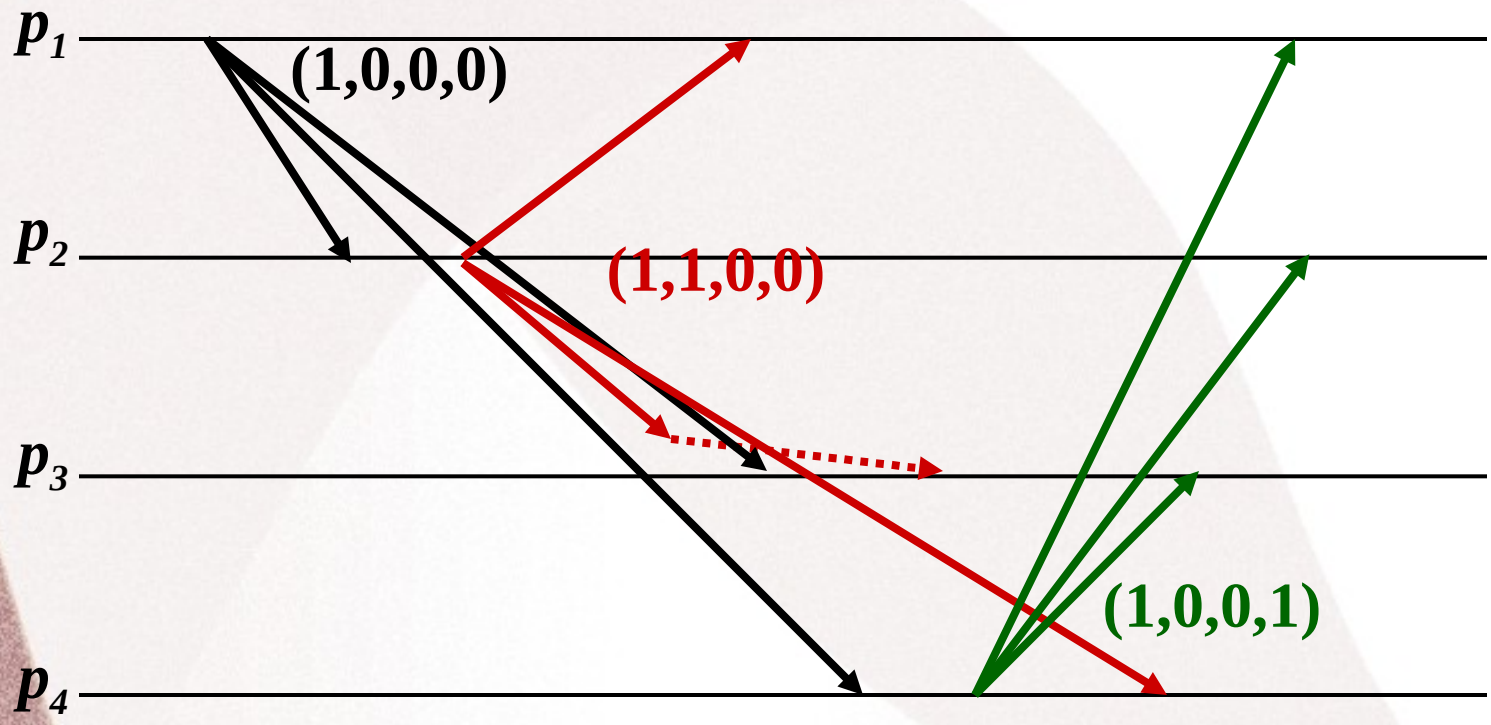
6.2. Protocolo CBCAST

- Sigue estos pasos:
  1. Antes de enviar *m*, el proceso p_i incrementa *RV(p_i)[i]* y marca *m*.
  2. Al recibir el mensaje *m* enviado por p_i y marcado con *RV(m),* los procesos p_j (distintos a p_i) retardan la entrega de *m* hasta que:

$$\forall k : 1..n \begin{cases} RV(m)[k] = RV(p_j)[k] + 1, & \text{si } k = i \\ RV(m)[k] \leq RV(p_j)[k], & \text{en otro caso} \end{cases}$$

  3. Cuando un mensaje *m* es entregado, *RV(p_j)* no se actualiza.
     - Esto contradice las reglas explicadas previamente para los relojes vectoriales, pero evita el rápido crecimiento de los valores de los relojes.
     - Sigue garantizando causalidad y mantiene todas las propiedades vistas previamente.

6.2. Protocolo CBCAST



6.3. Protocolo ABCAST

- Se utiliza un proceso coordinador (fijo), que dirige la ordenación. Se siguen estos pasos:
  1. El emisor difunde utilizando CBCAST el mensaje *m*, pero lo marca como no entregable. Los procesos que reciben este mensaje lo mantienen en sus colas, pero de momento no lo entregan, aunque pueda hacerse siguiendo las reglas de un CBCAST.
  2. El coordinador trata los mensajes ABCAST como si fueran CBCAST y los entrega siguiendo las reglas del CBCAST.
  3. Cuando el coordinador ha entregado uno o más mensajes ABCAST, utiliza CBCAST para difundir un mensaje SETS-ORDER con una lista de los mensajes ABCAST entregados.
  4. Cuando un proceso recibe un SETS-ORDER, lo interpreta y va entregando los mensajes ABCAST que aparecen en su lista, en ese mismo orden, y a medida que lo permitan las reglas del CBCAST.

6.4. Tolerancia a fallos

- El protocolo CBCAST puede no respetar la fiabilidad en la entrega si un emisor falla antes de haber enviado el mensaje a todos sus destinos.
- Para garantizar esta fiabilidad, cada nodo mantiene un contador que indica cuántos de sus mensajes difundidos han sido reconocidos por todos sus destinatarios (mensajes estables).
- En cada difusión realizada se añade al mensaje el valor de este contador.
- En caso de fallo, y antes de instalar la nueva vista, cada proceso comprueba si ha recibido algún mensaje “inestable” y, si es así, repite su difusión con su *RV(m)* original.

6.4. Tolerancia a fallos

- Cuando se han realizado todas las difusiones de los mensajes inestables, cada nodo difunde un mensaje FLUSH con el identificador de vista a instalar.
- Cuando un proceso ha recibido todos los mensajes FLUSH de todos los integrantes del grupo actual, se inician de nuevo las difusiones en la nueva encarnación del grupo.
- Al hacer esto, se ponen a cero todas las componentes de los relojes vectoriales.
- Desde que se detectara el cambio en el grupo hasta que termine el FLUSH, no puede difundirse ningún nuevo mensaje.