Universidad Nacional Autónoma de México

Facultad de Ciencias

Computación Distribuida

Tarea 12

Johann Ramón Gordillo Guzmán
418046090

Tarea presentada como parte del curso de **Computación Distribuida** impartido por la profesora **M.C Karla Rocío Vargas Godoy**.

02 de Febrero del 2021

Link al código fuente: https://github.com/JohannGordillo/

Índice

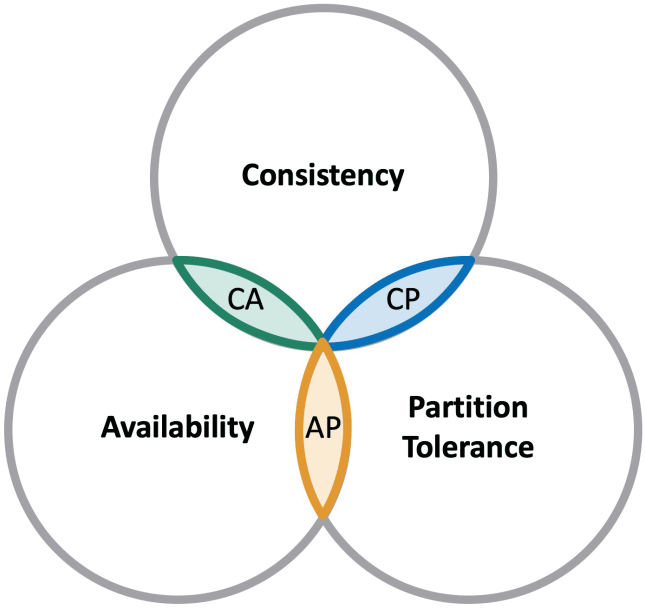
1 Teorema CAP

1. ¿Qué modelos de redes fueron utilizados para demostrar la conjetura de CAP hecha por Eric Brewer?

   Respuesta.

   El modelo de redes asíncronas. En este modelo no hay un relój y los nodos de la gráfica del sistema deben tomar decisiones únicamente con base en los mensajes recibidos y sus cómputos locales.

   Eric Brewer demostró que en el diseño de servicios web distribuidos es imposible cumplir al mismo tiempo las propiedades de disponibilidad, consistencia y toleracia a particiones. Es a lo que se le conoce como el Teorema de Imposibilidad CAP (Consistency, Availability y Partition Tolerance).



2. ¿Qué propiedades no se garantizan en el primer teorema (redes asíncronas)?

   Respuesta.

   El primer teorema nos dice que en redes asíncronas es imposible implementar un objeto de escritura y lectura de datos que garantice las propiedades de **disponibilidad** y **consistencia atómica** en todas las ejecuciones válidas, incluyendo aquellas en las que los mensajes se pierden.

   La propiedad de **disponibilidad** nos dice que cada petición recibida por un nodo correcto debe resultar en una respuesta del mismo. Una consecuencia es que las peticiones siempre terminan.

   La propiedad de **consistencia atómica** nos dice que debe existir un orden total en las operaciones, de tal manera que cada operación se ve como si hubiera sido completada en un único instante.

   Un corolario de este teorema es que en redes asíncronas es imposible implementar un objeto de escritra y lectura de datos que garantice la disponibilidad en todas las ejecuciones válidas y la consistencia atómica en las ejecuciones válidas donde no se pierdan los mensajes.

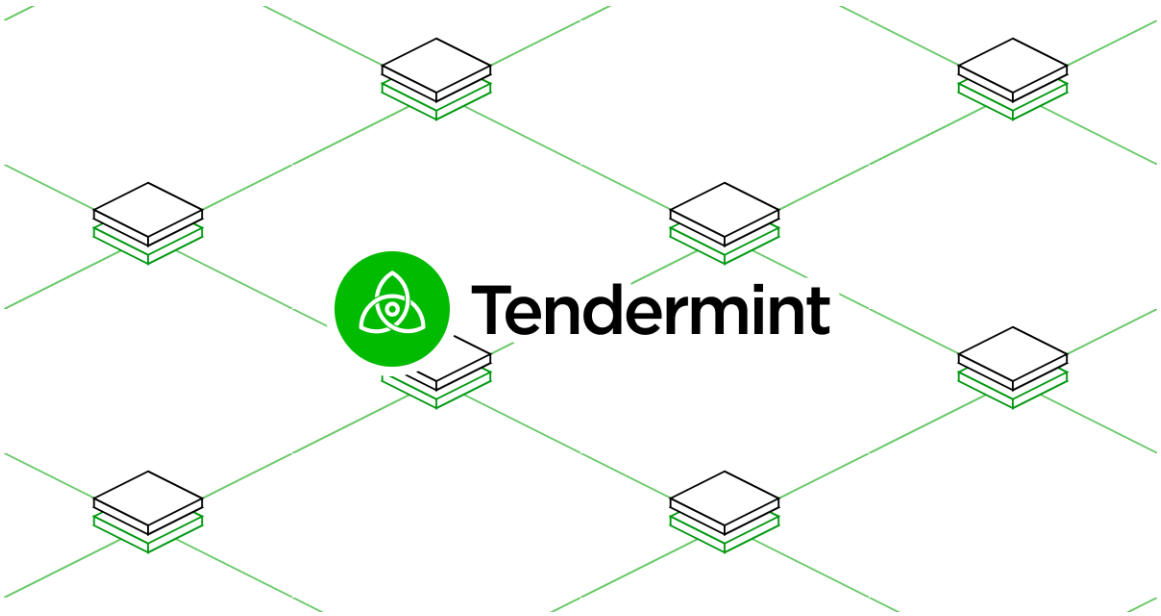
2 Sincronizadores y consenso bizantino

3. ¿Qué significa la propiedad de liveness, en algoritmos de consenso, y por qué algoritmos como HotStuff, Tendermint, etc., no garantizan esta propiedad en sistemas asíncronos?

   Respuesta.

   La propiedad de **liveness** nos dice que todos los procesos correctos eventualmente llegan a un acuerdo.

   Como el resultado de imposibilidad FLP nos dice que no existe un algoritmo para resolver el consenso bizantino que cumpla las propiedad de safety y liveness en sistemas asíncronos, es necesario sacrificar una de estas propiedades. Algoritmos como HotStuff, Tendermint, PBFT, etc. sacrifican la propiedad de liveness.



4. ¿Cuál es el objetivo de usar el sincronizador FastSync para resolver el consenso bizantino?

   Respuesta.

   FastSync es un sincronizador que puede ser usado por los algoritmos HotStuff, Tendermint, PBFT, etc. con el fin de garantizar la propiedad de liveness que éstos sacrifican, y de esta manera resolver el consenso bizantino en redes parcialmente síncronas, con restricciones realistas, donde exista un tiempo de estabilización global (GST) desconocido después del cual los delays en los mensajes enviados entre dos procesos correctos esté acotado por una constante δ desconocida.

   Así, el objetivo de FastSync es garantizar que después del GST todos los procesos correctos eventualmente coincidan en la misma vista (ronda en la que se encuentran los procesos) y que se queden en ella el tiempo suficiente para poder lograr el acuerdo.

   Algunas de las ventajas de este sincronizador es que maneja la pérdida de mensajes y que su implementación tiene latencia baja (el tiempo de respuesta es rápido).

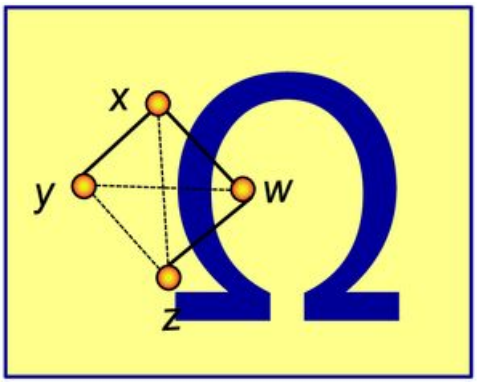
3 Leader crash recovery

5. ¿Qué diferencia existe entre el detector $\Omega_{crash/recov,omission}$ y Ω_{crash}?

   Respuesta.

   En el detector Ω_{crash} eventualmente todo proceso correcto siempre confían en el mismo proceso correcto.

   Por otro lado, en el detector $\Omega_{crash/recov,omission}$ eventualmente todo proceso correcto siempre confía en el mismo proceso correcto l y todo proceso incorrecto únicamente puede confiar en l o en ningún proceso.



6. ¿Qué tipo de procesos componen el conjunto CORE?

   Respuesta.

   CORE (o núcleo) es el conjunto de procesos eventualmente correctos del sistema distribuido. Los procesos eventualmente correctos son aquellos procesos que eventualmente permanecen activos para siempre.

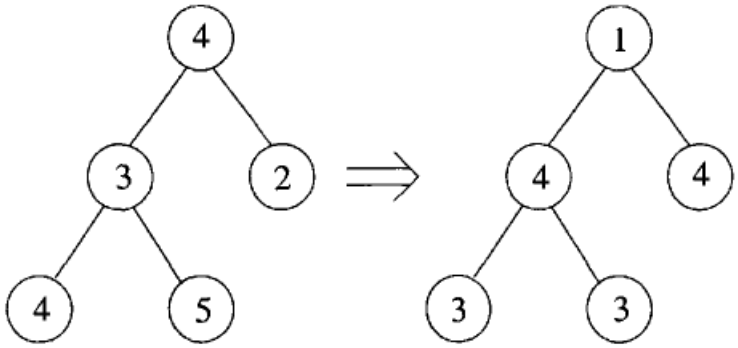
   A los procesos pertenecientes al conjunto CORE se les considera correctos y se asume que éstos son una mayoría en el sistema.

4 Vertex Coloring in log(n) time

7. ¿Por qué al recorrer los colores hacia abajo sobre un árbol se preserva la coloración?

   Respuesta.

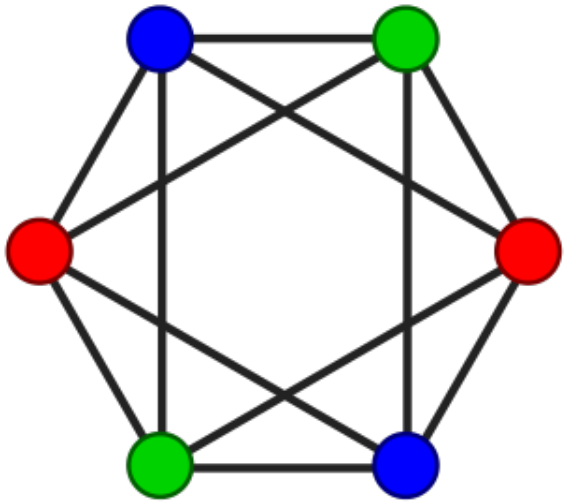
   Como los colores de la coloración original se recorren hacia abajo, dos vértices vecinos v, w tales que v es el padre de w, van a ser coloreados con colores distintos por como se define la operación shift-down y por ser la coloración original válida.



8. ¿Por qué es suficiente considerar $\Delta + 1$ colores disponibles para colorear un vértice?

   Respuesta.

   Sea G una gráfica con grado Δ. Consideremos un vértice v con Δ vecinos. Si todos los vecinos de v tienen colores distintos, para colorear a v con un color distinto a éstos necesitamos agregar un y sólo un nuevo color a la paleta de colores. Por lo tanto es suficiente considerar $\Delta + 1$ colores distintos para colorear un vértice.



5 Alternate Bit Protocol

9. ¿Qué mencionan las propiedades de Safety y Liveness?

   Respuesta.

   La propiedad de **safety** nos dice que *nada malo pasará*: En cualquier momento, la secuencia de elementos de datos procesados por el receptor R es un prefijo de la cinta de entrada X del emisor S.

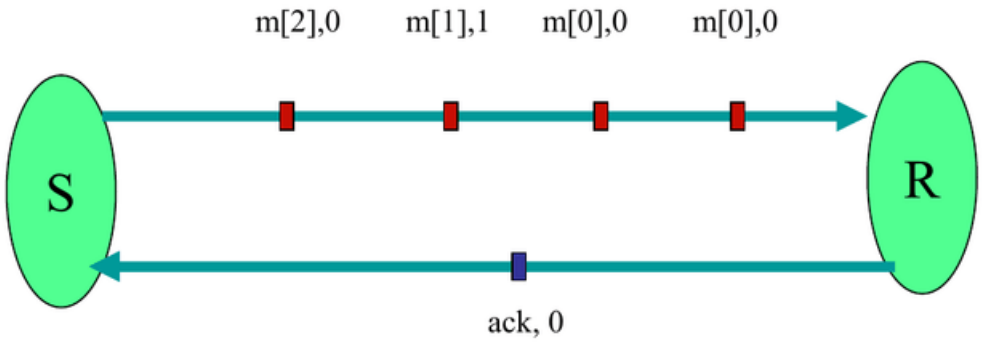
   Por otro lado, la propiedad de **liveness** nos dice que *eventualmente algo bueno pasará*: Todo elemento x_i de la cinta de entrada X del emisor S es eventualmente procesado por el receptor R, siempre y cuando el medio de comunicación cumpla las condiciones apropiadas de equidad (hay un equilibrio entre el ritmo en que se leen y se procesan los datos).

10. ¿Qué problema resuelve el protocolo?

    Respuesta.

    El Alternating Bit Protocol es una solución para el problema conocido como Problema de Transmisión de Secuencia (Sequence Transmission Problem) para canales FIFO asíncronos donde los mensajes pueden perderse, ser duplicados y corrompidos.

    Este algoritmo permite la correcta transmisión de información de un emisor a un receptor por medio de un canal no confiable. Es un caso particular del Sliding Window Protocol donde el tamaño de la ventana (número de datos que se pueden mandar) es 1.



6 Referencias

[1] Gilbert, S. & Lynch, N. (2002). *Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services.* ACM SIGACT News.

[2] Bravo, M., Chockler, G. & Gotsman, A. (2012). *Making Byzantine Consensus Live (Extended Version).* 34th International Symposium on Distributed Computing.

[3] Fernández-Campusano, C., Larrea, M., Cortiñas, R. & Raynal, M. (2014). *A distributed leader election algorithm in crash-recovery and omissive systems.* Elsevier, Information Processing Letters.

[4] Peleg, D. (2000). *Distributed Computing: A Locality-Sensitive Approach.* SIAM Monographs on Discrete Mathematics and Applications.

[5] Schmid, S. (2011). *From Shared Memory to Message Passing.* Technische Universität Berlin. Consultado en https://disco.ethz.ch/courses/podc_allstars/references/stefan/graph-coloring.pdf

[6] Piña, M. & Zenil, P. (s.f.). *The Alternating Bit Protocol.* Universidad Nacional Autónoma de México.