UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Tarea 1

INTEGRANTES

Torres Valencia Kevin Jair - 318331818
Aguilera Moreno Adrián - 421005200
Natalia Abigail Pérez Romero - 318144265

PROFESOR

Miguel Ángel Piña Avelino

AYUDANTE

Pablo Gerardo González López

ASIGNATURA

Computación Distribuida

6 de septiembre de 2022

1. (2 puntos) ¿Cuáles son las principales diferencias entre un sistema distribuido, un sistema concurrente y un sistema paralelo? Argumenta detalladamente.

Un sistema distribuido permite que una colección de computadoras independientes, que pueden o no estar ubicadas en distintos lugares, trabajar en conjunto para lograr una tarea. Utilizan protocolos especiales y paso de mensajes para coordinarse. La finalidad es es lograr solucionar problemas repartiendo el trabajo entre cada uno de los agentes del sistema.

En un sistema concurrente diferentes procesos o equipos se intercambian ya acceden a un mismo recurso durante periodos de tiempo separados de forma ordenada y nunca juntos. Este acceso puede ser tan rápido que desde el punto de vista del usuario pareciera que múltiples procesos acceden al recurso al mismo tiempo.

En un sistema paralelo se hace uso de 2 o más procesadores para resolver una tarea. Se basa en el principio según el cual, algunas tareas se pueden dividir en partes más pequeñas que pueden resolverse simultáneamente.

2. Retomando el problema de los dos enamorados con los mismos requerimientos vistos en clase, responda las siguientes preguntas:

- Suponga que las citas sólo se pueden realizar entre las 21:00 y las 22:00 horas. ¿Tiene solución el problema en este caso?

  El problema no tiene solución, porque a pesar de que el intervalo de tiempo en que la cita puede se realizada se vio reducido, persiste el problema de no poder reconocer cuando un mensaje ya ha sido entregado o se perdió, de manera que aún no es posible que los enamorados puedan ponerse de acuerdo para su cita.

- ¿El problema tiene solución cuando añade el siguiente requerimiento: los amantes deben ser capaces de coordinar una hora para una cita solamente cuando ningún mensaje se pierde, y, en cualquier otro caso, ellos no deberían presentarse?

  El problema no tiene solución, la imposibilidad es justo igual a la del problema original. Los enamorados no pueden saber si su mensaje ha sido recibido y si deben presentarse a la cita. Podría suceder que los mensajes se pierdan continuamente, de forma que los enamorados nunca se reunan.

- Consideremos una variación: Los dos amantes se han cuenta de que no necesitan ponerse de acuerdo sobre una hora exacta para la reunión, está bien si sus horas de reunión son lo suficientemente cercanas. En otras palabras, cada uno debería eventualmente elegir un tiempo, de modo que los dos tiempos estén lo suficientemente cerca. ¿Se puede resolver su problema?

  No, el problema no tiene solución. Porque es poco probable que dos intervalos sobre un periodo de tiempo indeterminado coincidan, además de que persiste el problema de indistinguibilidad, de forma que los enamorados tienen la incertidumbre si su mensaje fue recibido o no.

3. Investigue y explica brevemente el protocolo TCP. ¿Es posible resolver el problema de los dos amantes si hay un canal TCP confiable entre ambos amantes?

TCP es el Protocolo de Control de Transmisión que permite establecer una conexión y el intercambio de datos entre dos anfitriones. Este protocolo proporciona un transporte fiable de datos. La comunicación se realizaría en base a paquetes o segmentos.

Para establecer la conexión, es necesario que se reconozcan dos puntos de acceso y destino (cliente y servidor). Sin embargo, no importa qué parte hace de cliente y cuál hace de servidor. Lo único que necesita el protocolo TCP para establecer la conexión es que cada una de las partes cuente con una dirección IP y un puerto asignado.

Su prioridad es que los datos (segmentos) lleguen correctamente al destinatario, sin errores, y, en orden. Si en la transmisión de los segmentos, se corrompiesen o perdiesen, automáticamente el protocolo TCP inicia la retransmisión. De está manera, se garantiza que los datos lleguen al destinatario sin errores, ya que este protocolo se encarga de solucionar cualquier tipo de problema.

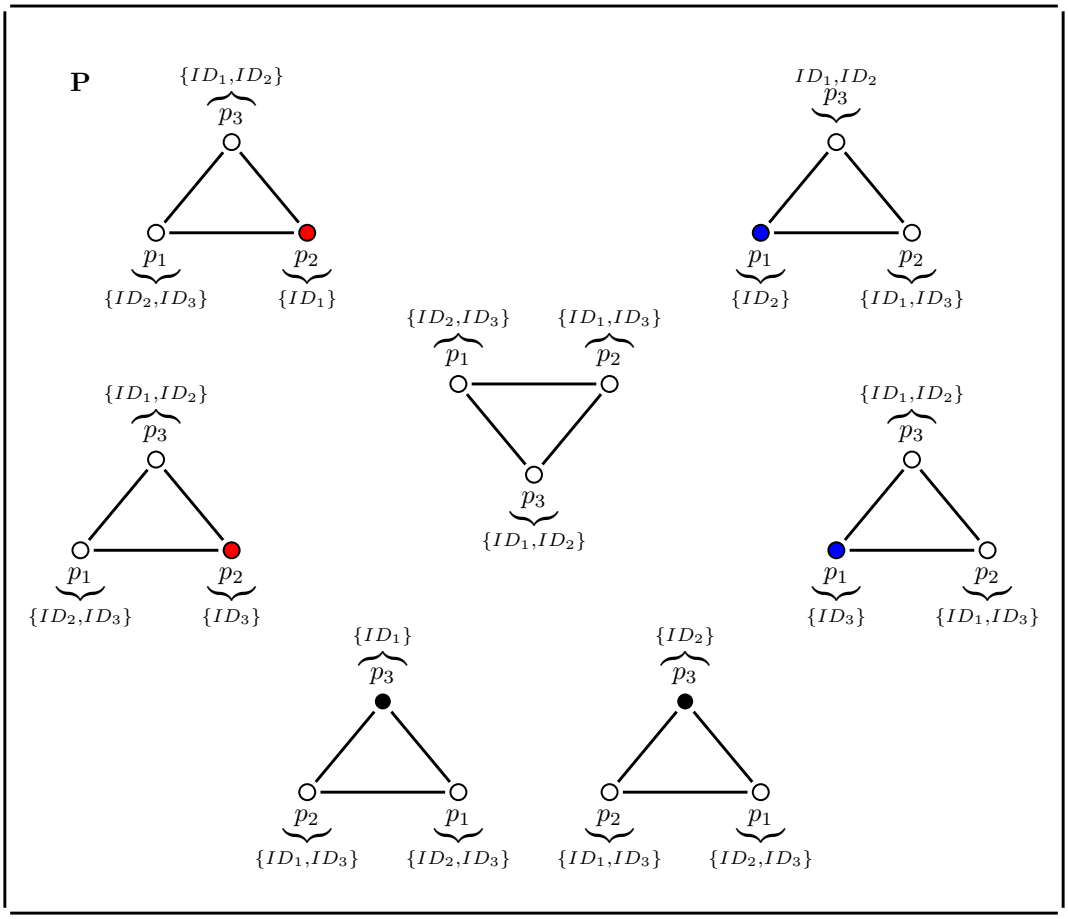
No es posible resolver el problema de los dos amantes si existe un canal TCP, solo nos garantiza una entrega confiable con un número infinito de reintentos/retransmisiones posibles. Conservandose, la prueba de imposibilidad del problema respecto al último mensaje que aún se mantiene. Por lo tanto, TCP no puede llegar a la misma decisión en ambas partes de la comunicación, como en el problema.

4. Considera un sistema distribuido con $n \geq 2$ procesos, $p_1, p_2, \cdots, p_n$, en el que la gráfica de comunicación es la completa K_n. El sistema es síncrono pero la comunicación no es confiable; sea P el conjunto de todos los procesos que envían mensajes en el tiempo d; entonces, hay dos posibilidades, todos los mensajes de P llegan a su destino en el tiempo $d+1$, o uno de ellos se pierde y nunca llega a sus destino y los otros en P si llegan en el tiempo $d+1$.

Considera un algoritmo A en el que cada proceso p_i tiene como entrada un identificador ID_i, que es un número natural (diferente al de los demás), y cada proceso p_i simplemente envía su ID_i a los otros $n-1$ procesos.

Dibuja cuales son todos los estados *globales* posibles (mundos posibles) en el tiempo 1 (los procesos mandan sus mensajes en el tiempo 0). En cada estado global, especifica el estado *local* de cada proceso, es decir, la información que cada proceso tiene en ese estado global; y entre cada par de estados globales pinta una arista con los procesos que no pueden *distinguir* entre esos estados. ¿Es posible que cada proceso elija consistentemente uno de los IDs de entre los que recibió de forma tal que en cada estado global todos los procesos eligen el mismo ID? Argumenta tu respuesta.

Solución: Supongamos que $n = 3$, entonces la representación gráfica de lo requerido se vería como el siguiente gráfico:



donde, a cada p_i $(1 \leq i \leq 3)$ le corresponde un ID_i de tal manera que para cada estado global se identifica si todos sus procesos fueron recibidos de manera exitosa, en caso de no ser así se colorean. Así, por cada proceso, tenemos que

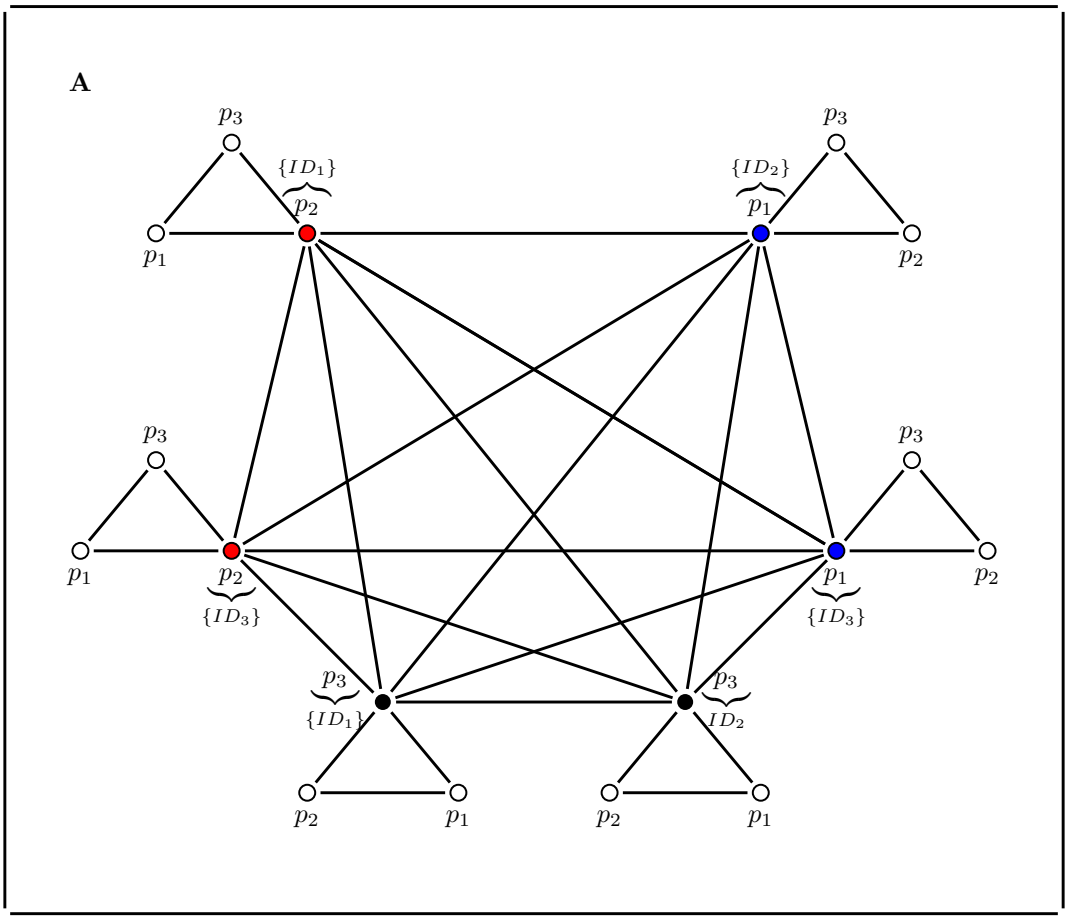
1. Sabemos que, a lo más, un mensaje no le llegó a p_1, o no le llegó a p_2, o no le llegó a p_3 (de manera exclusiva).

2. También sabemos que en el tiempo 0 cada proceso envía un identificador ID_i al resto de sus vecinos,

entonces, para un proceso p_i que no recibió un ID tenemos que existen dos posibles casos[I].

Obs. Los subíndices superiores o inferiores en cada gráfica nos dice que identificadores ID_i han recibido[II]. Además, se obvia que el proceso p_i ya tiene en su estado local al identificador ID_i.

A continuación mostramos un subconjutnto $A \subseteq P$, tal que cada proceso en el que no se pude distinguir, dentro del estado global, se conecten entre ellos



podemos observar que se ha ignorado la información no relevante para los procesos que no puede distinguir entre los demás estados.

Por último, demos respuesta a la pregunta: ¿Es posible que cada proceso elija consistentemente uno de los IDs de entre los que recibió de forma tal que en cada estado global todos los procesos eligen el mismo ID?
R./ No, esto es *falso* en general. Supongamos que en cada estado podemos elegir el mismo ID_i para todos los procesos, basta con ver que pasa para algún estado global en donde un proceso p_i no recibe exactamente el identificador ID_j[III] con $j \neq i$ y $1 \leq j \leq 3$.
Caso partícular con $n = 3$, es que todos los procesos eligan a ID_2 de manera consistente, claramente el estado global izquierdo superior no cumplirá esto, pues p_2 no ha recibido este ID_2. En este caso, hay más de un estado global que cumple no poder elegir de manera consistente este ID. Como hay al menos un proceso p_i en algún estado global que no recibió un ID_i, y esto pasa para cada ID_i, la respuesta a la pregunta es no.
Si solo existiera el caso ideal, donde todos los mensajes de P llegan a su destino, entonces la respuesta a esta pregunta sería sí. Bajo esta condición y bajo la consideración de que haya, al menos, un ID_i que haya logrado ser recibido por todos los procesos (exceptuando el proceso que envia), la respuesta sería sí. ◁

[I] Cada proceso tiene dos vecinos, por trabajar con K_3, en este caso partícular.
[II] El estado local del proceso.
[III] Este estado vive en P.

5. Tomate 10 minutos de tu tiempo y ve el siguiente video: The Man Who Revolutionized Computer Science With Math. Quanta Magazine. Presenta un breve reporte de lo que trata dicho video.
Leslie Lamport es un computologo, incluso antes de que fuera identificado como tal, él cambio su manera de pensar los programas y los concibió más como un objeto matemático, algo que puede ser demostrado así se dio cuenta de que diseñaba algoritmos.
Pues un algoritmo que no puede ser demostrado no es más que una conjetura, y el probar corresponde a las matemáticas.
Los computologos tienden a pensar en términos de lenguajes de programación y suelen confundir programación (programming) con codificación (coding). Pero los programas están construidos en ideas por lo que no deben estar limitados por un lenguaje de progrmación, así estos tienen que hacer algo y transmitir esta idea antes de codificarlos.
Es por esto por lo que Leslie Lamport decidió crear un lenguaje (TLA+) que fuese utilizado para diseñar, modelar, documentar y verificar programas.
En el 2013 Leslie Lamport gano el Premio Turing, el cual es un premio de las Ciencias de la Computación otorgado anualmente por la Asociación para la Maquinaria Computacional (ACM) a quienes hayan contribuido de manera trascendental al campo de las ciencias de la computación, por sus contribuciones en los sistemas distribuidos, donde múltiples componentes en diferentes redes se coordinan para lograr un objetivo.
Lamport nos dice “Un sistema distribuido es uno donde tu computadora puede terminar inutilizable por una computadora que no sabías que existía”. Mientras la computación no distribuida diferentes procesos se comunican usando la misma memoria, en la distribuida se comunican usando mensajes.
Lamport se interesó en la computación distribuida cuando llego a sus manos un manuscrito de Paul R. Johnson y Robert H. Tomas, quienes trabajaban en un algoritmo para implementación de bases de datos distribuidas, estas bases de datos tienen múltiples copias en diferentes computadoras para que los programas de estás tuvieran rápido acceso a los datos.
Lamport explica en su artículo como con la noción de causalidad (causality) nos da la posibilidad de resolver cualquier problema de computación construyendo una máquina de estados (state machine). La cual podemos pensarla como una computadora abstracta que lidia con una cosa a la vez: asegurarse que todas las computadoras en el sistema distribuido colaboren para implementar una sola máquina de estados.
El algoritmo de la panaderia (bakery algorithm) utilizado para implementar la exclusión mutua, es decir, evitar que dos procesos usen la impresora al mismo tiempo. Utiliza números para identificar los procesos, pero lo que lo vuelve realmente es especial es que no requiere hacer suposiciones.