Universidad Zaragoza
1542

Shamuel Manuel Manrrique Aquino
NIP: 802400

Practica 1: Sistemas Distribuidos
Fiabilidad, Causalidad y Entrega Grupal

Resumen

El siguiente informe describe brevemente las decisiones de diseño e implementación de la práctica de la asignatura Redes y Sistemas Distribuidos usando el lenguaje de programación **go**, la cual consiste en la implementación de una librería paso de mensaje que garantice fiabilidad, causalidad y entrega grupal.

Objetivos generales:

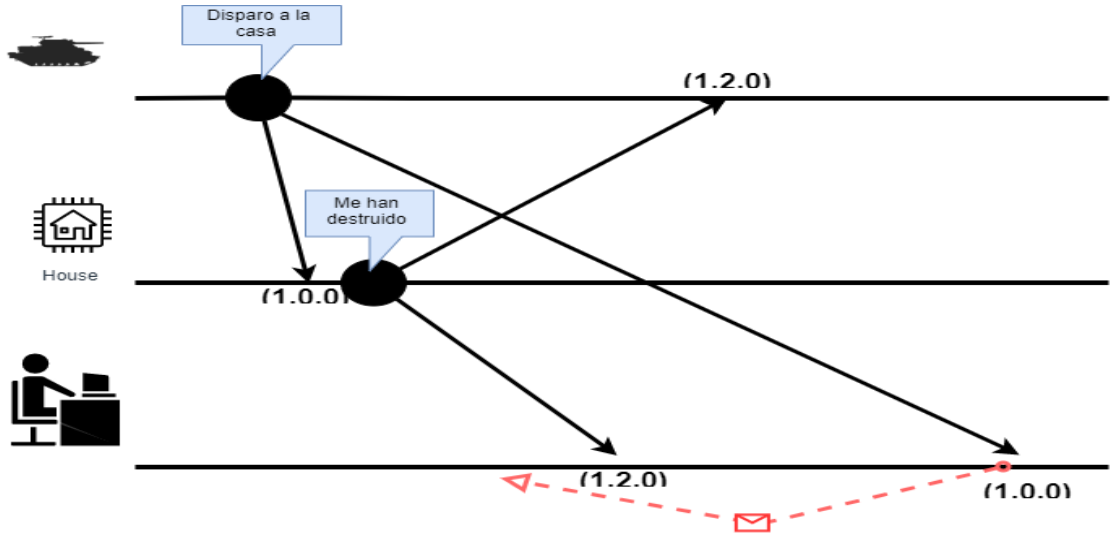
1. Comprender cómo se comunican y manejan los eventos los sistemas distribuidos.
2. Conocer las ventajas y desventajas que ofrecen los protocolos UDP y TCP/IP en aplicaciones distribuidas, de acuerdo a latencia, tolerancia fallo, retardos en la red y entrega de paquete.
3. Entender los mecanismos de sincronización en sistemas distribuidos y la captura de estado globales consistentes en un sistema distribuido.
4. Conocer la comunicación por ip multicast sus beneficios y particularidades.

Implementación

La presente práctica estuvo dividida en tres secciones:

1. **Comunicación de Grupo Causal:** Se implementó una librería que usa el protocolo TCP que consta de cuatro funciones(send, receive, sendGroup y receiveGroup) que se encargan de la comunicación entre procesos y causalidad usando la librería vClock. Para la explicación del funcionamiento de este módulo de la librería se expone el caso de un juego que contiene tres procesos en un determinado escenario donde el primer proceso **P1** es un **tanque**, el segundo **P2** una **casa** y el tercero **P3** un **observador**. En este ambiente podemos analizar la propagación de mensajes entre los distintos procesos que se ejecutan usando el protocolo **TCP/IP** por lo que me da garantía de secuenciación, fiabilidad y demás ventajas de este protocolo, pero no me garantiza causalidad por lo que introducimos los relojes vectoriales como elemento que nos permite diferenciar cual es la causalidad de la secuencia de eventos que ocurren internamente. Dado que es un juego que se están ejecutando en un escenario donde los tres se ven y los tres juegan, cualquier mensaje puede influenciar en cualquier hecho que ocurra en ese escenario, en algunos casos eso les incide y otros no, eso qué quiere decir que en la propagación de evento al resto como

acción individual, los demás no participan. A Continuación se muestra una gráfica donde se detalla lo descrito anteriormente.



En la imagen superior podemos ver los tres procesos descritos anteriormente y como el P1 (Tanque) inicia una serie de eventos disparando a P2 (Casa) he informando a el resto de proceso (P3) a P2 recibir el mensaje del cual resulta ser víctima notifica a los demás que P1 le ha disparado. Problemática: como se puede observar en el gráfico el P3 (Observador) recibe primero el mensaje de P2 por problemas en la red en ese punto se ve la importancia de los relojes vectoriales para el tratamiento de la causalidad, el proceso P3 al percatarse del problema existente en la recepción de los mensajes usa los relojes vectoriales para mostrarle al observador los eventos en orden cronológico. Es importante resaltar que esta implementación con el uso de TCP los envíos se ejecutan secuencialmente uno a uno el problema es que si tengo mucho procesos a quién enviar se generan retardos considerables en la red cosa que no queremos dado que se ven afectados los procesos, este proceso constituye lo que se llama envío de grupo. Por ende a nivel de implementación tanto en un juego como en otros entornos es un problema la latencia al notificar los eventos.

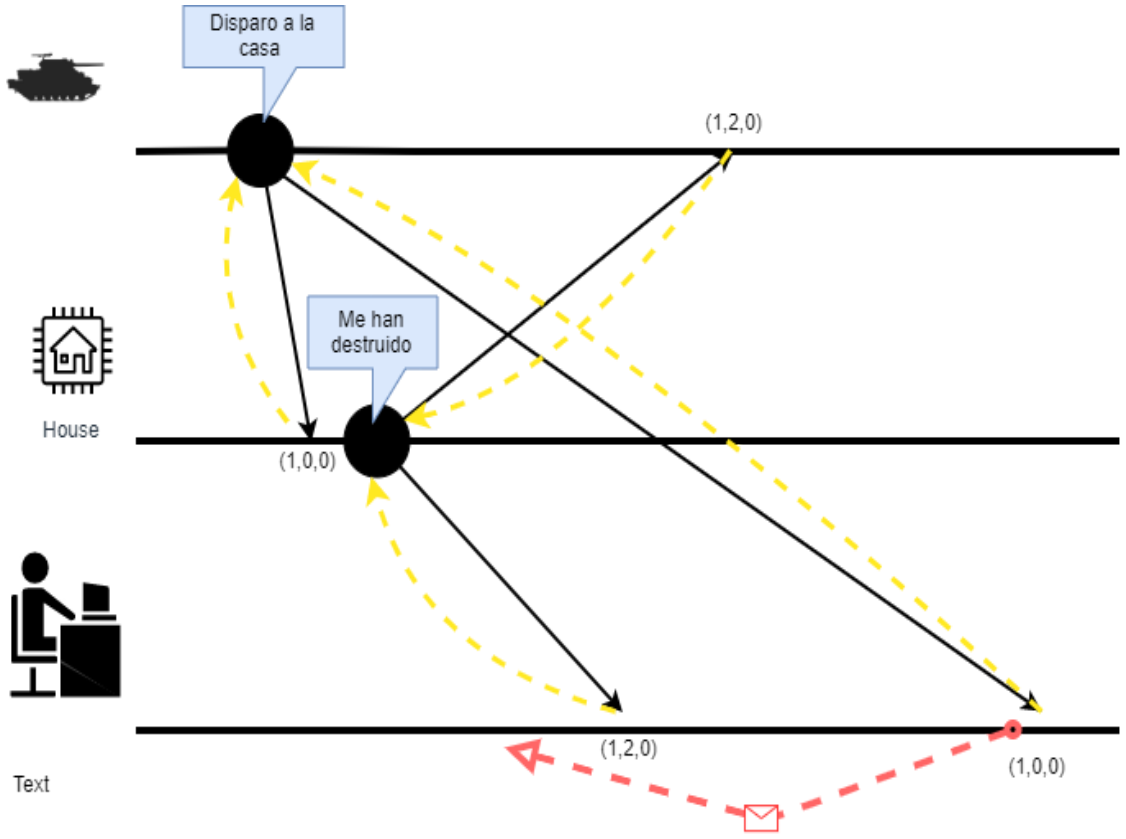
Universidad Zaragoza
1542

Shamuel Manuel Manrrique Aquino
NIP: 802400

2. **Comunicación de Grupo Causal fiable con IP Multicast:** Dada la problemática planteada en el módulo anterior, en pro de plantear una mejorar para esta situación se introduce en el segundo módulo de la librería ip multicast (es un direccion ip de clase D que permite con solo enviar un mensaje a dicha dirección y todos los procesos que están escuchando de ese ip reciben el mensaje inmediatamente) es decir, una propagación tecnológica que nos permite enviar mensaje a todos los que pertenecen al grupo multicast simultáneamente. El uso del protocolo UDP permitió ganar tiempo en la propagación de mensaje sacrificando la fiabilidad que garantiza TCP y tampoco contamos con garantías causales. Es por eso que se realizaron consideraciones extras como al momento de realizar un envío debe recibir un Ack (confirmación) del proceso que recibió el mensaje, de no ocurrir esto se hace un envío unicast usando UDP. Así se amplía el protocolo de comunicación para que cada proceso respecto al origen tenga que confirmar la recepción del mensaje mediante ACK que es básicamente lo que hace TCP. Por otra parte para añadir causalidad simplemente usamos reloj vectorial que permite guardar las garantías causales entre los eventos que ocurren en sistemas distribuidos entre diferentes procesos. La recepción de las confirmaciones(ACK) y reenvíos de mensajes en este módulo se hacen punto a punto. Al hacer un envío de mensaje de punto a punto te quedas esperando la confirmación, si no te llega reenvias nuevamente un número determinado de veces. En caso contrario finaliza la ejecución con error en la comunicación. En este caso el delay es manejado en él receive(en caso que se hiciera en el send todos reciben el delay, comportamiento que se quiere evitar), que es como ocurre realmente en la red detalle del que me percate en este segundo módulo, dado que para el primero simulaba retardos en red antes de enviar.

   La ventajas de esta implementación es la optimización de los recursos, dado que envío un solo mensaje al router y este envía el misma a quien le interesa. El RFC5771 contiene el manual completo con la información de direccionamiento multicast las direcciones ip de clase D son las usadas para direccionamiento multicast estas están reservado generalmente, se tiende a usar todos los host 224.0.0.1 al 224.0.0.255 para el control de bloqueo de área local los host pueden responder a la dirección del router

Shamuel Manuel Manrrique Aquino
NIP: 802400



En la imagen superior se puede observar el mismo ejemplo planteado en el módulo la diferencia es que este refleja unas flechas amarillas que salen del punto de la recepción del mensaje que no son más que el ACK de recepción. En este módulo se decidió enviar el mensaje tres veces a la dirección ip multicast (cada uno de estos envíos tienen un pequeño retraso respecto al anterior). Al un proceso recibir un mensaje este envía un ACK directamente por unicast al proceso que envió el mensaje, en caso de que el proceso emisor no recibe los ACK completos envía a los procesos que faltan por confirmación el mensaje por unicast usando UDP, luego de este proceso si aún falta algún ACK el programa termina por que no podemos garantizar fiabilidad, en caso contrario continua con la ejecución.

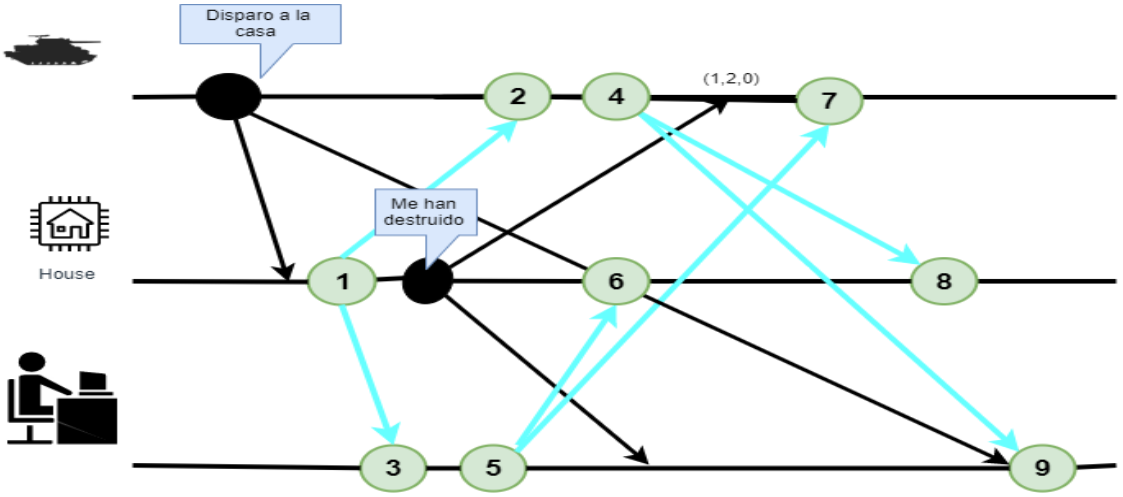
Shamuel Manuel Manrrique Aquino
NIP: 802400

3. **Algoritmo Chandy-Lamport para determinar estados globales en sistemas distribuidos:** En este módulo se implementa el algoritmo de Chandy-Lamport con el objetivo de obtener un estado global consistente que se construye a partir de una instantánea del sistema. Lo ideal sería hacerlo de forma precisa, es decir solicitando a cada uno de los procesos que guarden su estado en un instante de tiempo, pero no es posible sincronizar a la perfección todos los relojes en un sistema distribuido por lo tanto es imposible crear un instantánea precisa. Lo que sí podemos hacer es tomar una instantánea que denominaremos consistente, no es precisa por que se va a tomar en diferentes instantes de tiempo pero al menos será coherente, por lo que no reflejaremos ninguna entrega de mensaje que no esté precedida por su respectivo envío.

   Para la implementación de este algoritmo necesitamos garantizar fiabilidad entre los canales de comunicación por lo que se usa la implementación de Comunicación de Grupo Causal desarrollada usando TCP. El desarrollo de esta sección cuenta con una estructura de tipo Marker que permite a cualquier proceso al recibir la orden para iniciar un snapshot luego de esto debe grabar su estado(variables globales hasta ese punto en el Header de la estructura Marker e inicializa su checkpoint (contadores o count) sabiendo que la grabación terminará cuando haya recibido n-1 contadores, donde n es el número total de procesos. Luego de esto envía al resto de procesos los checkpoint), cada vez que se recibe un contador se guarda los mensajes en un arreglo que llamado Channel(los arreglos no son más que una un almacén de grabación dado un canal). Contamos con sistema distribuido donde existe un canal de comunicación entre cada par de procesos(un canal de P1-P2 y otro P2-P1) y garantizamos que el canal es fiable(TCP) la transmisión de mensajes se hace en orden FIFO (First In First Out) esto para la creación de instantánea consistente del estado global del sistema distribuido.

Shamuel Manuel Manrrique Aquino
NIP: 802400

A Continuación se muestra un gráfico y una tabla explicativa donde se ilustra y explica la ejecución de Chandy-Lamport y la grabación de un estado global consistente



Estado Local	Count	Cabecera	Chanel	
P1	2	Recibo el contador de P1 veo que el primero que me llega, guardo estado mi estado (incluye el mensaje m1 que envíe) y envío el marcador.	p2 Count 2	Vacío
			p3 Count 7	Mensaje m1
P2	1	Guardo estado de P2 (incluye el mensaje m1 que recibió) e inició la Grabación.	P1 Count 8	Vacío
			p3 Count 6	Vacío
P3	3	Recibo el contador de P3 veo que el primero que me llega, guardo estado de P3 envío el marcador.	p1 Count 3	Mensaje m2
			p2 Count 9	Vacío