REDES Y SISTEMAS DISTRIBUIDOS
Prof Unai Arronategui

Mini Proyecto de Simulación distribuida

Francisco Martinez #794893

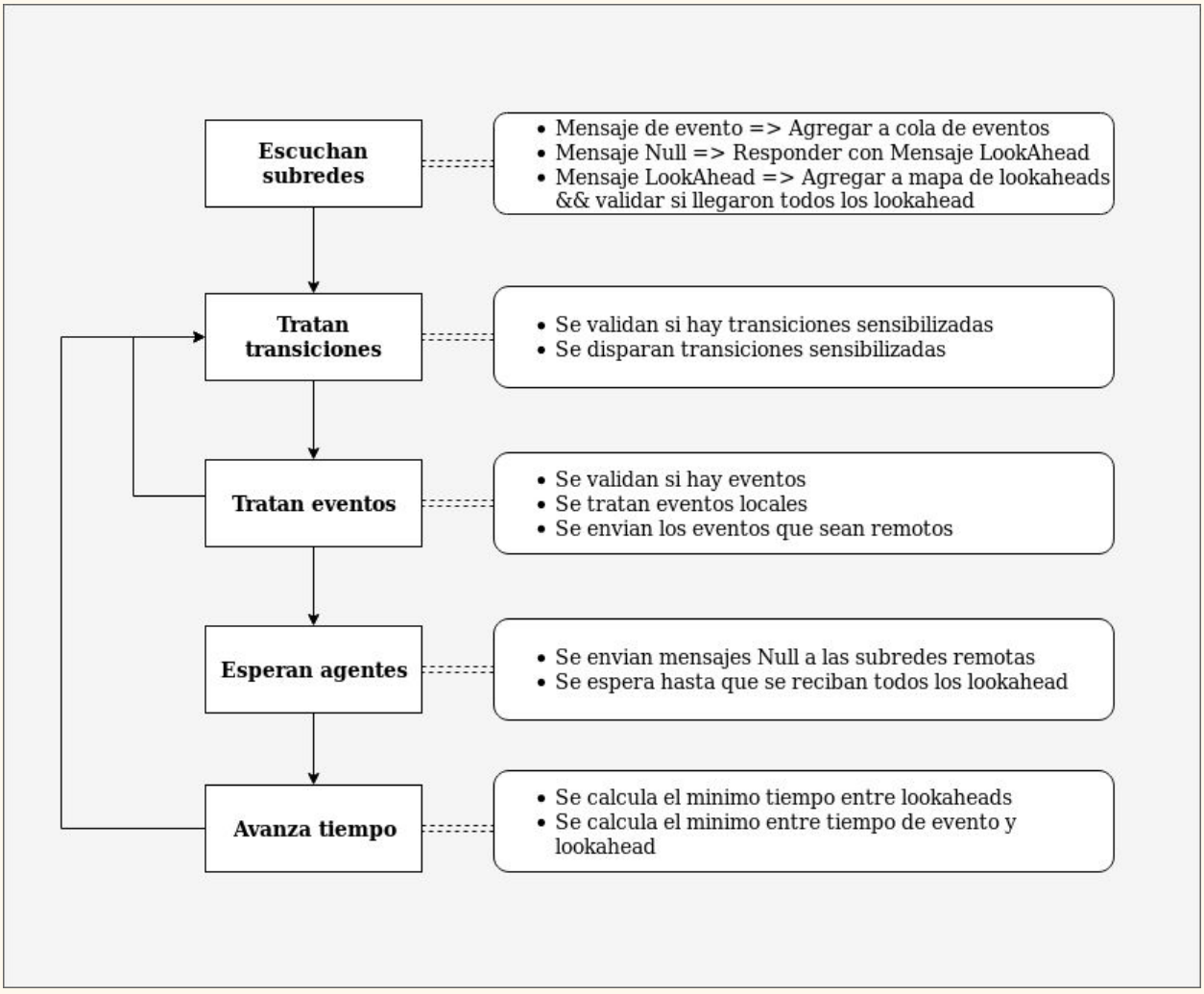
INTRODUCCION

Este trabajo plantea el diseño e implementación de un simulador distribuido de Redes de Petri. La clase de redes de Petri a simular son las binarias (1 solo token por lugar) y sin conflictos (solo una transición como sucesor a cualquier lugar). La tarea parte con un código base de un simulador de redes de petri local el cual será expandido para poder contemplar la distribución de la red de petri en distintos nodos de una red de computadoras, a través de la memoria a continuación hablaremos de el diseño de alto nivel de la solución, los protocolos de interacción y detalles de implementación asociados.

DISENO DE ALTO NIVEL

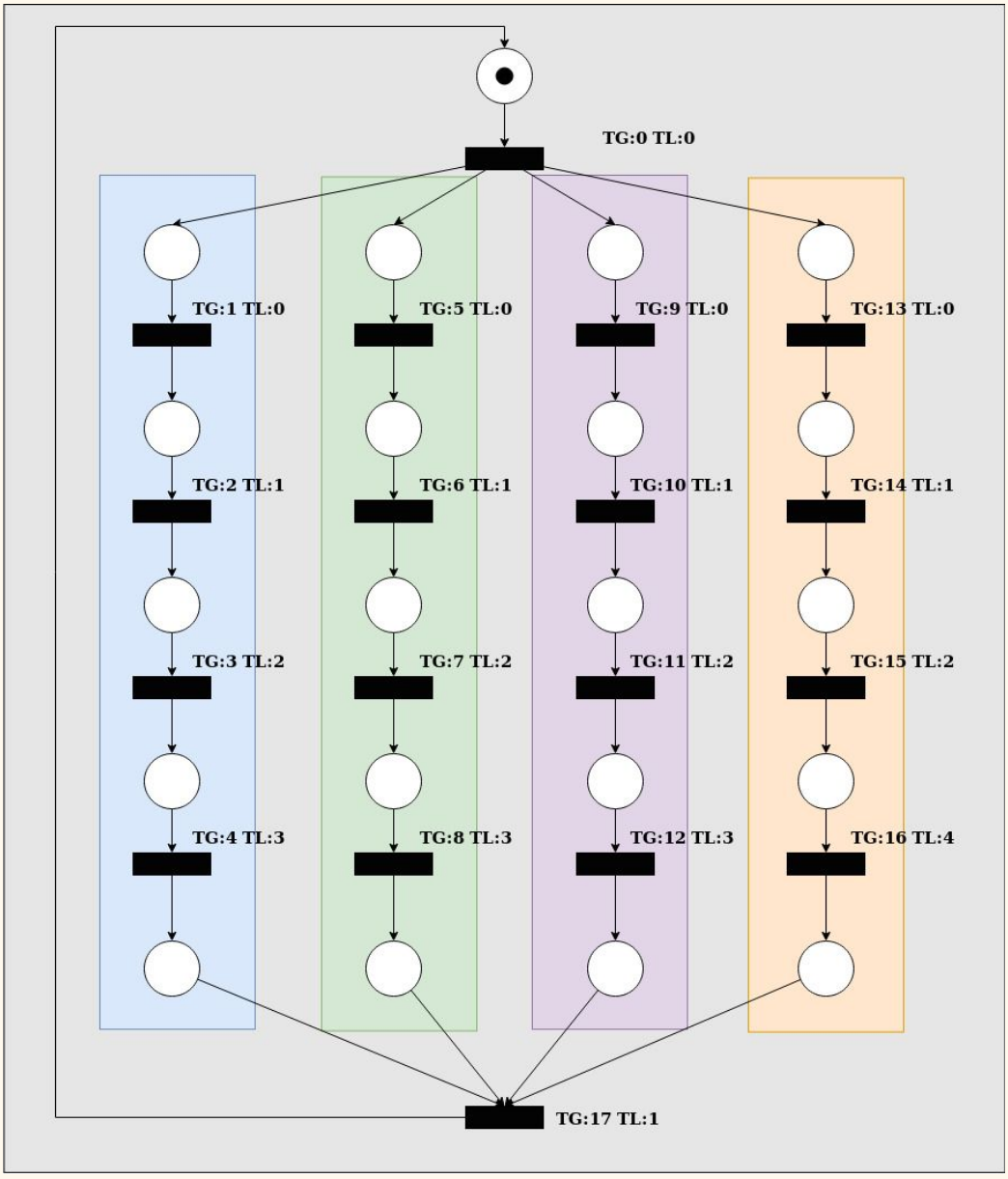
En este gráfico se define a grandes rasgos el proceso de simulación distribuida que se implementó donde cada nodo de la red distribuida que lo integra sigue este conjunto de pasos y lógica.

Se comienza con una rutina que se encarga de **escuchar los distintos mensajes** que se pueden recibir con el protocolo TCP que están encriptados en GOB, si este mensaje es de tipo ***Evento*** entonces estos eventos serán traducidos a un evento local donde la transición a afectar se traducirá del índice global recibido, si el mensaje en cambio es de tipo ***Null*** simbolizando la necesidad de un mensaje LookAhead de alguna subred entonces se le responderá con dicho valor el cual es la suma de el reloj local mas la duracion de disparo de la transición pertinente y finalmente si el mensaje que se recibe es de tipo ***LookAhead*** este se agregara a un mapa que hace seguimiento de los mensajes recibidos de este tipo hasta que todos los mensajes esperados lleguen. Luego se **trata las transiciones** lo cual comprende identificar si existen transiciones sensibilizadas y luego dispararlas para así crear nuevos eventos, posterior a este paso se **procesan los eventos** ya sea manipulando transiciones locales o enviando los eventos a través de la red a otras subredes en caso de que estos requieran procesamiento remoto. Estos últimos pasos exceptuando el de crear una rutina de escucha concurrente se repiten hasta que ya no existan transiciones sensibilizadas localmente pero para mantener una simulación conservativa de la red de petri es necesario pedir mensajes de LookAhead a través del envío de mensajes Null para saber hasta qué punto es posible avanzar el reloj de la simulación local. El último paso de la simulación sería avanzar el tiempo de nuestro reloj local el cual lleva el ritmo de nuestra



simulación distribuida, para esto se calcula el mínimo tiempo entre todos los mensajes de lookahead y el tiempo mínimo de evento que tengamos(si es que existe), estos pasos se repiten hasta alcanzar el tiempo de simulación objetivo.

A continuación mostraremos la red de petri simulada y sus particiones serán resaltadas en colores distintos para su apreciación, en este gráfico se usarán TG para denotar el nombre global de la transición mientras que TL sera el nombre local de la transición:



DETALLES DE IMPLEMENTACIÓN

Se realizará una breve explicación de las principales funciones manipuladas y expandidas además de detallar las peculiaridades de implementación y sus implicaciones:

```
func (self *SimulationEngine) Simular(ai_cicloinicial, ai_nciclos
TypeClock)
```

Esta función fue manipulada para arrancar ***listen_subnets()*** que se encarga de realizar la escucha de los mensajes de red. También se agrego la funcion ***esperar_agentes()*** que envía y espera mensajes lookahead, un último detalle es que Simular espera 10 segundos antes de comenzar la ejecución para dar tiempo de prender todos los nodos de la simulación distribuida.

```
func (self *SimulationEngine) listen_subnets()
```

Esta función recibe mensajes utilizando el protocolo TCP y maneja los distintos casos de mensaje posible los cuales son ***Evento, LookAhead, Null.***

```
func (self *SimulationEngine) tratar_eventos(ai_tiempo TypeClock)
```

Esta función fue modificada para enviar el evento en caso de que el índice de transición al cual esté asociado sea negativo, de esta manera simbolizamos que un evento es remoto las ventajas de esto fue una implementación más simple pero crea el problema de que la transición con nombre 0 no puede ser identificada correctamente si es remota(en nuestro caso ninguna remota le envía a la que posee de nombre 0). No es posible comenzar las transiciones desde 1 ya que el funcionamiento del simulador dado utiliza el número como posición de la lista de transiciones al momento de acceder. Una alternativa era un booleano en la estructura de transición.

```
func (self *SimulationEngine) esperar_agentes()
```

Esta función envía mensajes de tipo Null a cada subred que está definida como una precondición de nuestra Lef local, esta estructura es un mapa donde tenemos como clave el nombre de la transición global y de valor su dirección de escucha en la red. Posteriormente se bloquea hasta que todos los mensajes de lookahead de cada subred remota hayan sido recibidos, para esto chequea un booleano en la estructura de motor de simulación que es colocado como True por ***listen_subnets()*** cuando se reciben todos los lookahead.

```
func (self *SimulationEngine) avanzar_tiempo() TypeClock
```

Esta función fue manipulada para que al momento de calcular el tiempo al que puede avanzar el reloj local de la simulación tomará en cuenta los valores de lookAhead recibidos después de haber enviado y esperado por los agentes. Este valor será el mínimo entre los valores, y el valor lookahead enviado por cada subred representará su tiempo de simulación local sumado a la duración de disparo de la transición que se encargue de disparar el evento.

```
func (self *SimulationEngine) calculate_la(idGlobal IndLocalTrans)
TypeClock
```

Esta función se encarga de buscar la duración de disparo dado el identificador global de la transición que fue recibido en el mensaje ***Null*** correspondiente para luego sumarle el valor de nuestro tiempo de simulación local el cual será enviado en el mensaje de ***LookAhead***.