Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC3272 - Criptomonedas y contratos inteligentes

Proyecto Consensus from Trust

22 de mayo de 2019

1er semestre 2019 - Profesor D. Vrgoc

Maximiliano Santamaria — José Tomás Vigneaux

Repositorio: `https://github.com/jtvigneaux/proyecto-1-criptomonedas`

1. Teoría General Consenso distribuido

Según la definición que vimos en clases, en un consenso distribuido todos los participantes del sistema están de acuerdo sobre las transacciones válidas. Si no están de acuerdo, las transacciones no van a aparecer en el Blockchain (en realidad van a aparecer por un tiempo, pero esa rama no se va a seguir expandiendo).

Es el método para lograr la descentralización, ya que no existe una entidad decidiendo qué mensaje es válido, sino que los participantes de la red lo deciden al estar todos de acuerdo. Por eso mismo todos los nodos tienen que tener el Blockchain descargado.

En Bitcoin el consenso funciona usando mining via proof of work. Los nodos están siempre escuchando o recibiendo nuevas transacciones. Las agrupan en un bloque y buscan el nonce para poder publicar el siguiente bloque en la red. Los otros nodos van a aceptar el bloque si es que es válido, es decir si el consenso distribuido es correcto. Luego, si es aceptado, se extiende el Blockchain incluyendo el bloque reciente.

2. Consensus from Trust en nuestro proyecto

El protocolo de nuestro consenso no usa mining via proof of work, sino que es aleatorio si un nodo puede proponer un bloque y todos tienen la misma probabilidad pp de proponer un mensaje. Por lo mismo, en un caso hipotético, no existirían recompensas para los nodos que proponen mensajes, ya que todo sería aleatorio.

Además, el grafo es dirigido, en donde los arcos indican la confianza que existe entre los nodos. Si un nodo A confía en un nodo B, significa que el nodo A va a aceptar cualquier transacción que reciba del nodo B. De esta forma se simula si un mensaje es válido. La red funciona en base a la confianza que tengan los nodos entre ellos, por lo que es posible deducir que un grafo con bajo nivel de confianza puede dejar nodos aislados que nunca van a estar en consenso.

En el grafo pueden existir nodos maliciosos, al igual que en Bitcoin y los nodos normales pueden confiar en estos nodos maliciosos aceptando sus mensajes, lo que significa un problema de seguridad para la red. Además, los nodos maliciosos nunca expanden mensajes recibidos desde otros nodos.

3. Modelación

3.1. Clases

Las clases ocupadas para la modelación del problema fueron:

`Transaction`

Esta clase esta basada en el enunciado del proyecto. Tiene un tipo, que está siempre fijo, un valor, que va a cambiar dependiendo de la transacción, y un uniqueID.

`Node`

Los nodos tienen un set de transacciones, un ID y el tipo (si es malicioso o bueno).
Además, los métodos que tiene esta clase son:

- `is_malicious()`: Retorna un `bool` de si el nodo es malicioso o no.
- `malicious_strategy()`: Retorna la estrategia maliciosa del nodo y 0 si no lo es.
- `get_transactions()`: Retorna las transacciones que están guardadas en el nodo.
- `print_transactions()`: Imprime cada una de las transacciones que están guardadas
- `check_transaction(transaction)`: Retorna un `bool` de si la transacción está en el nodo.
- `add_transaction(transaction)`: Añade al set de transacciones del nodo la transacción dada.

`Graph`

Clase hecha para juntar todos los nodos y relacionarlos entre ellos. Tiene una lista de nodos y un diccionario con las conexiones de cada nodo.
Los métodos de esta clase son:

- `get_nodes()`: Retorna la lista de nodos del grafo.
- `get_connections()`: Retorna el diccionario de conexiones del grafo.
- `get_ammount_of_nodes()`: Retorna cantidad de nodos del grafo.
- `add_connection()`: Añade una conexión, si esta no existe, entre dos nodos.
- `propagate_message(node, transaction, firts=True)`: Función recursiva que propaga el el mensaje a todos sus conexiones.
- `show_graph()`: Mostrar graficamente el grafo

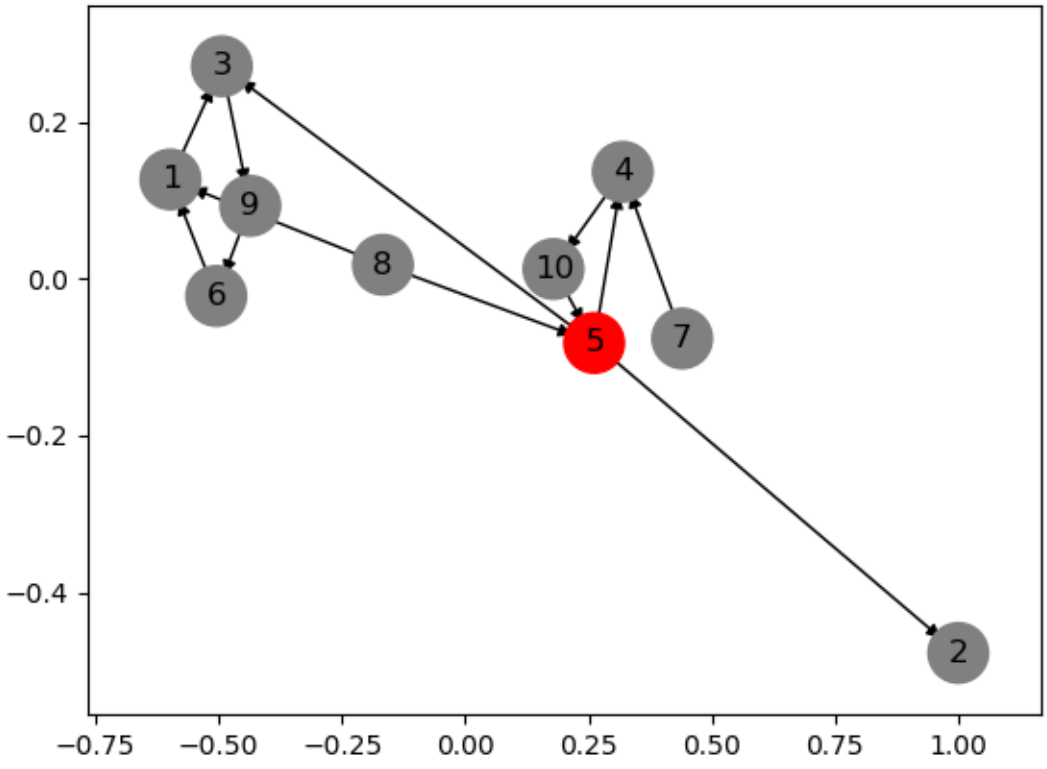

Figura 1: Grafo hecho con 10 nodos, probabilidad de conexión de 16 % y de nodos maliciosos de 10 %

`Simulator`

Esta clase toma como parámetros un grafo, las probabilidades p, pp y ppp, el número de iteraciónes de la simulación y los guarda en sus atributos.
Los métodos de la clase son:

- `generate_transaction()`: Retorna una instancia de la clase transacción generada aleatoriamente.
- `nodes_in_consensus()`: Retorna la cantidad de nodos que hay en consenso.
- `print_nodes_info()`: Imprimir la información de cada uno de los nodos.
- `statistics()`: Sacar la media y desviación estándar de los datos.
- `show_consensus()`: Mostrar graficamente el número de nodos en consenso versus el tiempo que lleva la simulación.
- `run()`: Ejecuta toda la simulación con los parámetros dados.

3.2. Funciones

Además de las clases descritas anteriormente también se ocuparon funciones. Estas fueron:

- `GenNetwork(n, p, pp)`: Genera un grafo al azar en base a los parámetros dados. (cantidad de nodos, probabilidad de realación y probabilidad de ser malicioso).
- `multiple_runs(iterations, number_of_nodes, p, pp, ppp, k)`: Ejecuta múltiples ejecuciones de la simulación con los mismos parámetros.
- `evolution_changing_parameter(iterations, number_of_nodes, p, pp, ppp, k, parameter)`: Genera gráficos de las simulaciones realizadas.

4. Resultados y Análisis

Los siguientes resultados se obtienen midiendo cómo cambia el consenso en un grafo al cambiar los valores de los parámetros del problema (conectividad, tasa de nodos maliciosos, número de rondas, número de nodos, etc.)

Para eso se partió desde un escenario estándar con 10 nodos, 60 % de probabilidad de conectividad, 50 % de probabilidad de recibir un mensaje nuevo, 10 % de probabilidad de ser malicioso y 10 rondas. Luego, se elegía algún parámetro y se generaba un escenario distinto al variarlo. Por ejemplo si queríamos medir la evolución con respecto a la probabilidad, se probaban los escenarios con conectividad $\in \{0; 0{,}1; 0{,}2; 0{,}3; 0{,}4; 0{,}5; 0{,}6; 0{,}7; 0{,}8; 0{,}9; 1\}$. Para

cada uno de estos escenarios se hacen 50 iteraciones y se saca el promedio y finalmente se grafica el promedio de consenso.

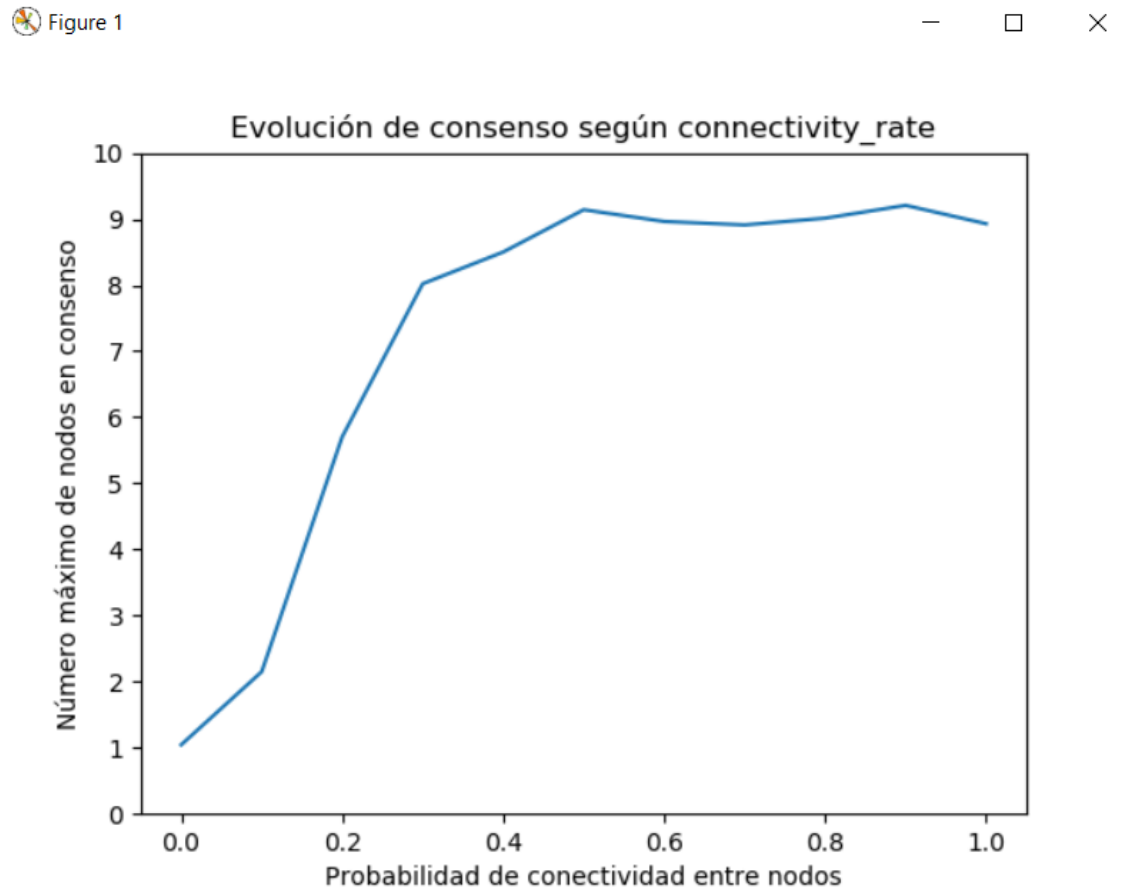

Figura 2: Consenso vs conectividad

Se puede ver que a mayor conectividad en el grafo, existe mayor consenso. La razón para esto es que si hay baja conectividad, significa que los nodos no confían entre ellos, por lo que no van a escuchar los nuevos mensajes (los van a tomar como inválidos) y pueden quedar nodos aislados. Pero si aumenta la confianza, la red se va conectando y los nodos se expanden correctamente. Es interesante notar que es necesaria una probabilidad de 0.4 para alcanzar el nivel de consenso más alto, ya que después de esto se mantiene el consenso en un mismo nivel. Por lo tanto no es necesario gastar tanto recurso en tener una red 100 % conectada, sino que con un 40 % basta.

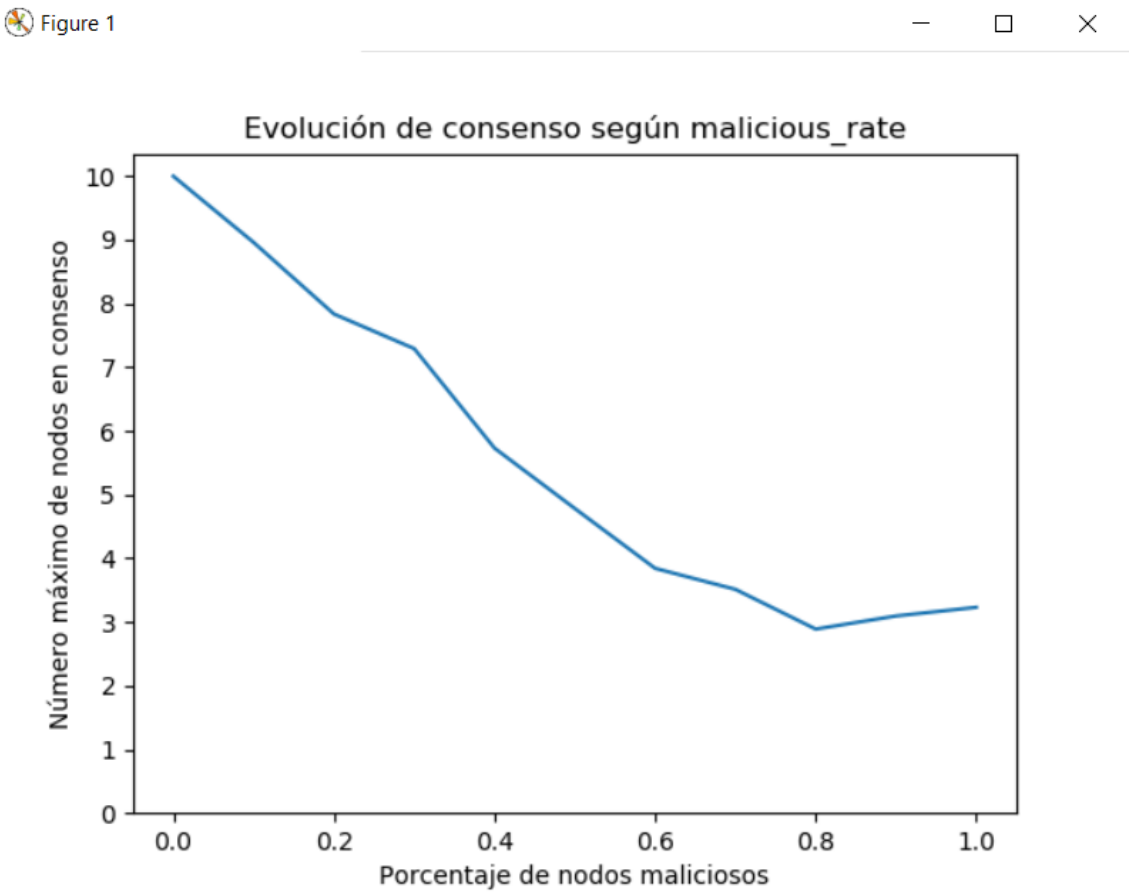

Figura 3: Consenso vs nodos maliciosos

A mayor porcentaje de nodos maliciosos, disminuye el consenso. Esto es porque los nodos maliciosos no siguen expandiendo un mensaje válido y generan diferencias entre las listas de transacciones. Aparte, el consenso llega hasta un mínimo de 3 nodos de acuerdo, ya que si hay muchos nodos maliciosos, algunos nodos nunca van a recibir transacciones y van a mantener su lista vacía (es decir, van a estar en consenso que no existen transacciones).

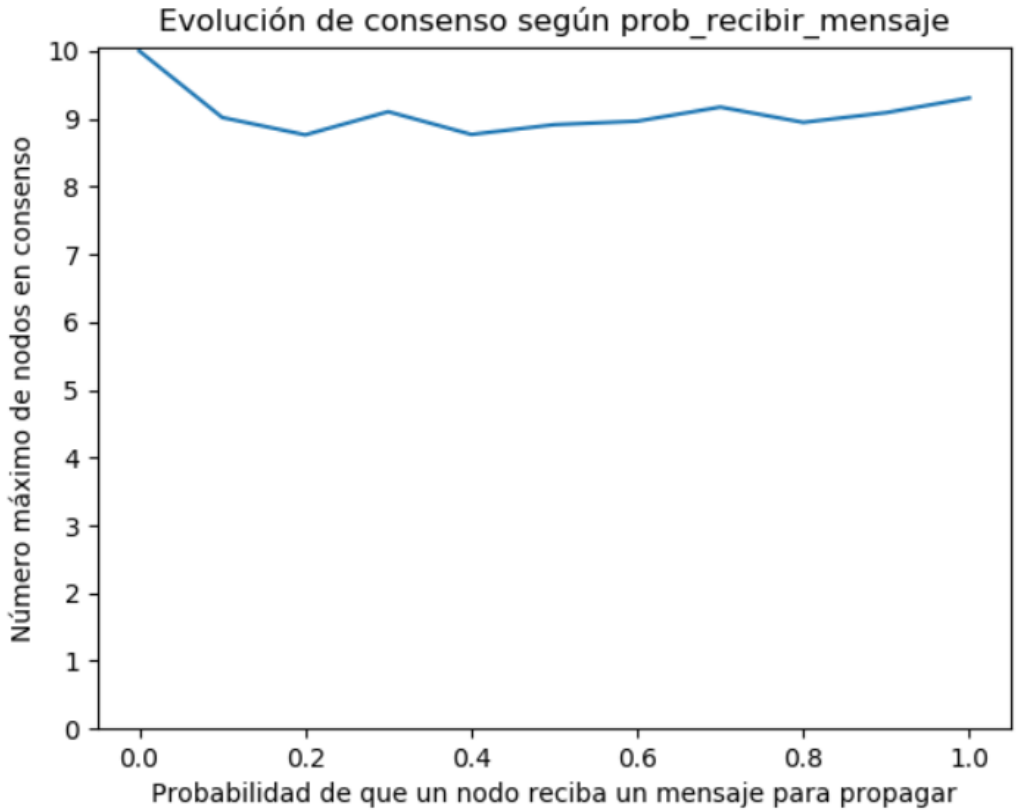

Figura 4: Consenso vs puntos de propagación

Este gráfico muestra que no es tan relevante la probabilidad de que un nodo reciba un mensaje para propagar, ya que el nivel de consenso se mantiene estable. Esto es porque basta con que un nodo reciba el mensaje para que este se pueda expandir, en caso de que el grafo esté bien conectado y no hayan lugares aislados.

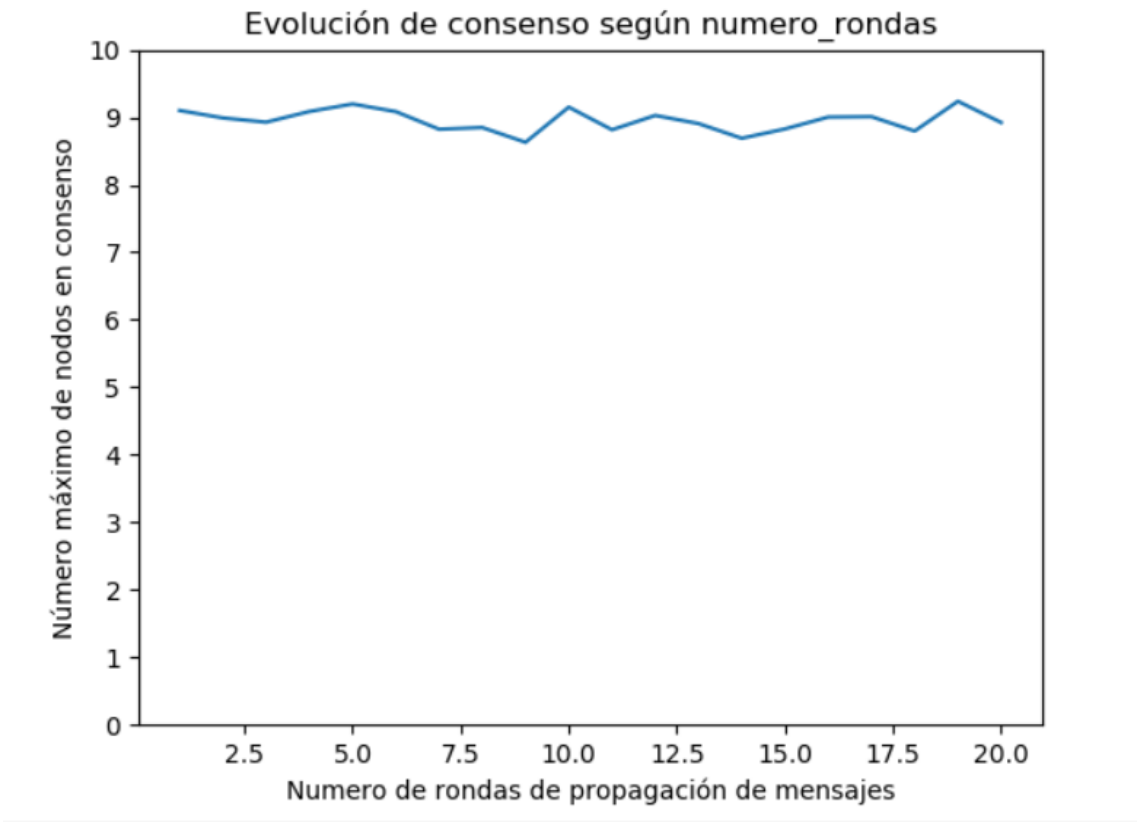

Figura 5: Consenso vs rondas

Nuevamente vemos un comportamiento estable del consenso al aumentar el número de rondas de la simulación. Esto es otra evidencia para decir que la conformación y estructura del grafo es lo que importa para el consenso y no la cantidad de mensajes que se propaguen. Una red bien formada, puede tener un correcto funcionamiento del consenso en un período muy largo.

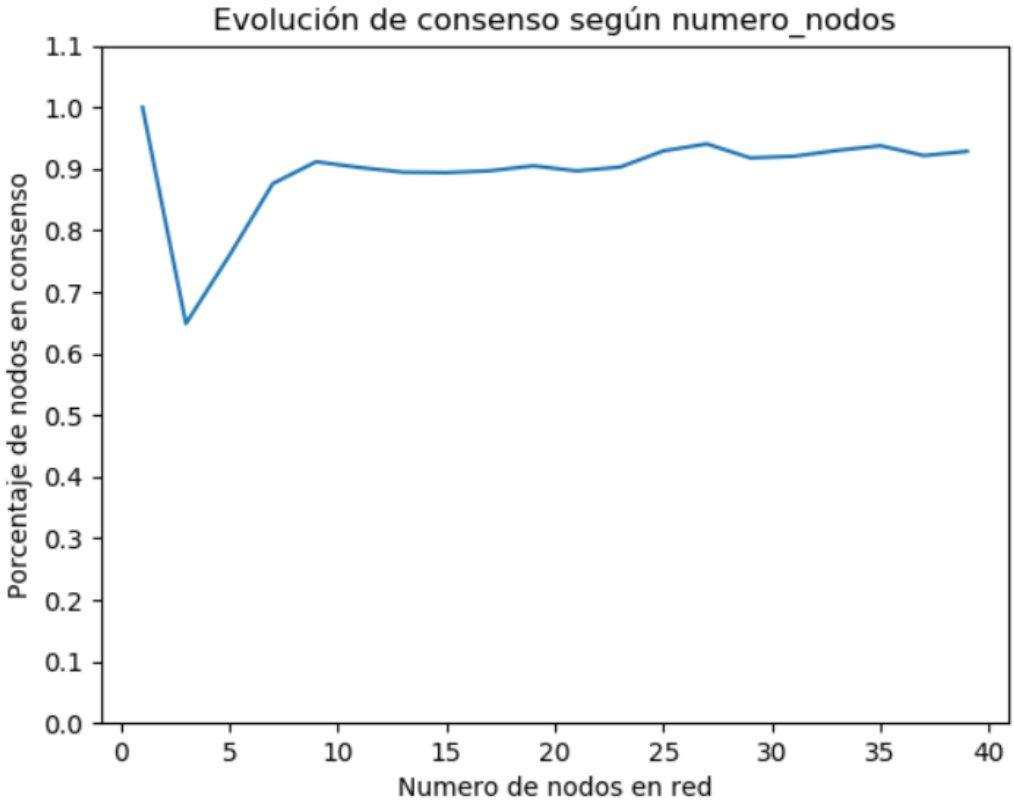

Figura 6: Consenso vs nodos

Finalmente intentamos expandir la red a una mayor cantidad de nodos y medir esta vez el porcentaje de consenso, lo cual es la cantidad de nodos en consenso dividido en el total de nodos en la red. Se aprecia que de igual forma se propaga bien el mensaje, sin importar el aumento de tamaño. Al principio con una baja cantidad de nodos hay menos consenso, ya que basta con que un nodo no coincida para que se note en la proporción total, pero a partir de 10 nodos el porcentaje de consenso es estable y no influye el cambio del valor del parámetro.

Este parámetro llamó la atención al hacer pruebas cambiándolo en combinación con los factores de conectividad y nodos maliciosos (los cuales se detectaron como importantes anteriormente), ya que se comprobó que a mayor cantidad de nodos, se podía lograr el consenso en la red con menor conectividad. Esto es porque si hay más nodos, existen también mayor cantidad posible de arcos entre los nodos, por lo que no es necesario una probabilidad tan alta de confianza, ya que van a existir otros caminos que expandan el mensaje. Lo mismo para los nodos maliciosos, en una red más grande podemos tener más nodos maliciosos sin que afecte al consenso porque van a existir rutas alternativas que compartan el mensaje que un nodo malicioso detiene y no propaga. El consenso en total disminuye con más nodos

maliciosos, pero es porque los nodos que no están de acuerdo son precisamente los maliciosos, no los nodos normales.

5. Conclusiones

El trabajo nos sirvió para entender de manera más profunda el funcionamiento del consenso distribuido y los factores que Satoshi tuvo que considerar para que funcionara en su sistema descentralizado.

Luego de implementar y ejecutar nuestra simulación, nos dimos cuenta que algunos parámetros influían mucho más que otros en el consenso que se podía lograr en una red. Lo primero que notamos es que un grafo ideal es aquel que está completamente conectado y no tiene nodos maliciosos. En ese caso todos los miembros están de acuerdo con las transacciones en todo momento. Pero al ingresar los nodos maliciosos había de inmediato nodos que no iban a mantener la misma lista de transacciones, ya que estos nodos no aceptaban los mensajes que recibían (a menos que fueran ellos quienes propusieran el bloque). Hicimos varias pruebas cambiando el porcentaje de maliciosos y encontramos un patrón, que era que el consenso disminuía si se aumentaban los maliciosos.

Siguiendo este análisis, buscamos patrones al cambiar los otros parámetros dentro de la simulación y concluimos que el consenso no depende del tamaño de la red, ni de la cantidad de mensajes que se reciban, sino que depende principalmente del grado de conectividad entre los nodos y de la cantidad de nodos maliciosos. Si se logra encontrar un valor óptimo para estos 2 parámetros, la red puede funcionar perfectamente según el consenso para grandes cantidades de nodos y mensajes.

Finalmente si se pidiera proponer un valor óptimo para la red, depende del tamaño de esta. Por ejemplo para un tamaño de 10 nodos, nosotros eligiríamos una conectividad del 40 %, ya que sobre este valor no hay mayor ganancia en consenso. Como se explicó en resultados, a mayor cantidad de nodos no se necesita tanto porcentaje de conectividad, ya que hay más conexiones posibles por lo que este valor que entrega el máximo consenso va a variar. También se buscaría el menor valor posible de nodos maliciosos (cercano al 10 % sería óptimo), pero que de ninguna forma supere el 50 %.