

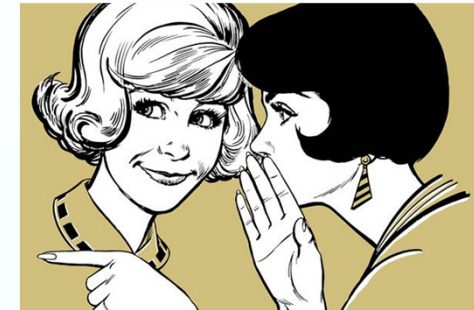
Sistemas Multiagente

Práctica: Ejemplo Spade

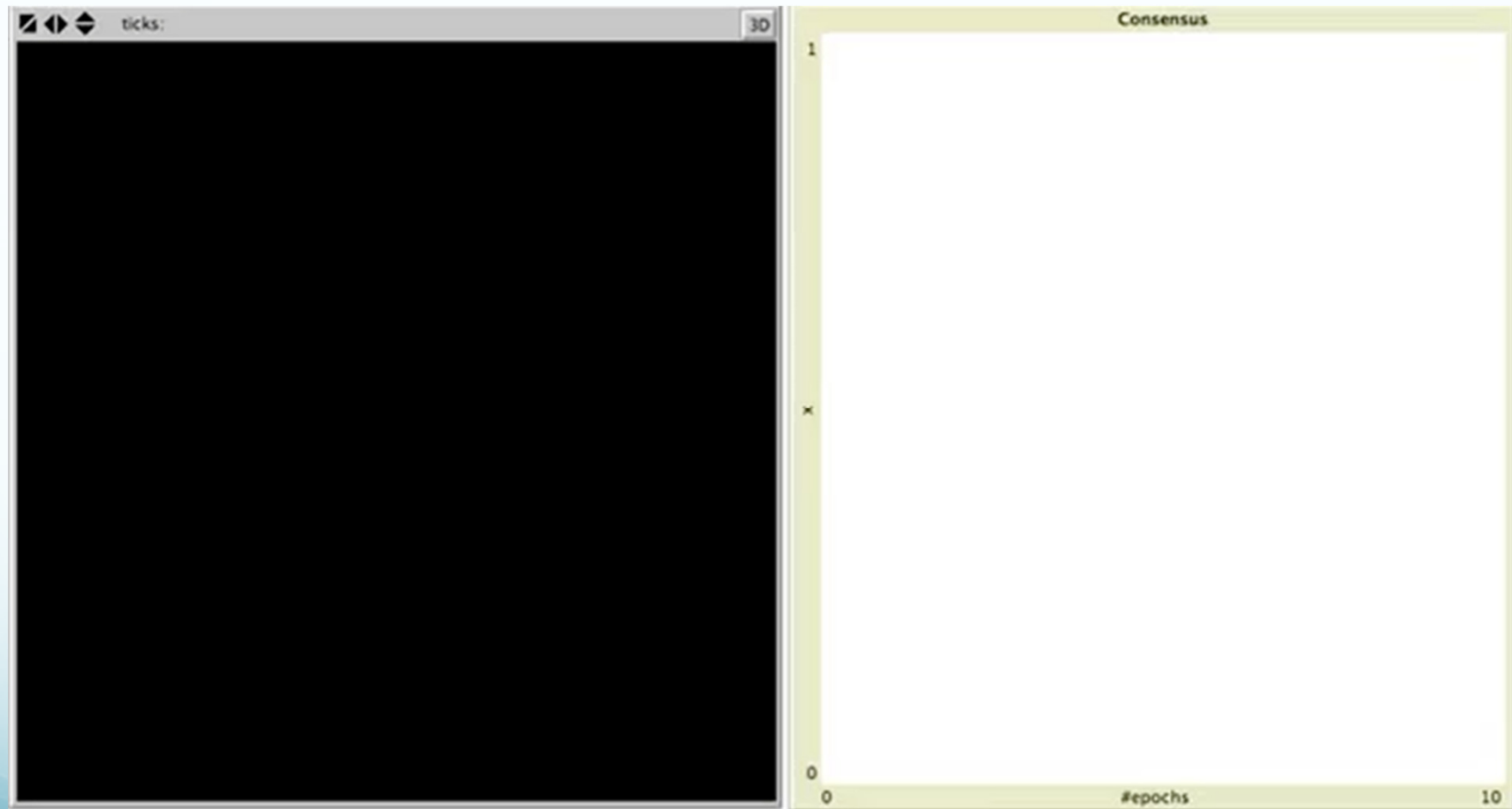
Vicent Botti, Vicente Julián

Comunicación: Gossiping

- En un algoritmo Gossip, cada nodo de una red intercambia información periódicamente con un subconjunto de nodos.
- Este subconjunto suele ser el conjunto de vecinos de cada nodo
 - Cada nodo sólo tiene una vista local de la red
- Objetivo: cada nodo recibe la información global deseada, a través de un cierto número de actualizaciones periódicas de los nodos.
- Aplicación: difusión en redes de sensores, coordinación de equipos de robots, análisis de la propagación de virus, de noticias falsas.....



Comunicación: Gossiping

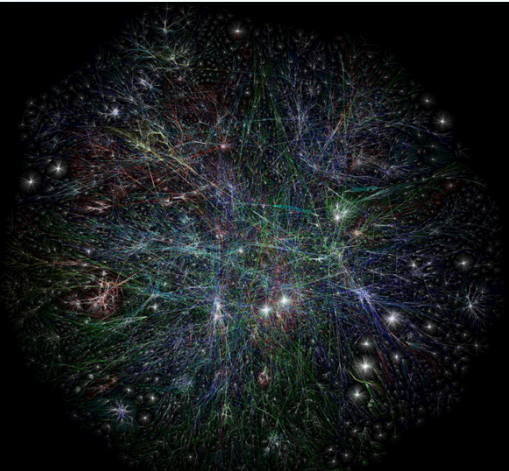


Comunicación: Gossiping

- **Problema**
 - Diseñar un algoritmo para que todos los agentes de un sistema reciban un rumor lo más rápido posible.
- **Solución I:** El agente inicial envía el rumor a todos sus vecinos, y cada agente informado lo reenvía a su vez a todos sus vecinos.
 - cada agente necesita interactuar con todos sus vecinos.
 - cada agente recibe múltiples copias del rumor.
- **Solución II:** Construir un árbol de expansión y transferir el rumor sólo a lo largo del árbol.
 - Fallos de comunicación en enlaces del árbol puede romper el proceso de propagación de rumores.

Comunicación: Gossiping

Solución alternativa:

- Necesitamos un algoritmo simple, ejecución local, distribuido, rápido y robusto para la difusión de la información.
 - Protocolos Push, Pull, Push-Pull
- 
- Hay un rumor inicialmente localizado en un agente de una red (y todos los agentes deben compartir información)
 - El protocolo procede por rondas, en las que cada agente sólo contacta con uno de sus vecinos (o con un subconjunto).

Comunicación: Gossiping

Protocolo Push

- Visión simple del algoritmo desde el emisor
 1. $t=0$
 2. mientras que $t < T$ hacer
 3. cada agente informado envía el rumor a un vecino aleatorio.
 4. $t=t+1$

Propiedades:

- Los nodos sólo contactan con sus vecinos; la estructura global de la red es desconocida para cada nodo.
- Robustez: El fallo de transmisión entre unos pocos nodos no afectará al rendimiento del algoritmo.
- El algoritmo envía eficientemente un rumor a todos los nodos de la red.

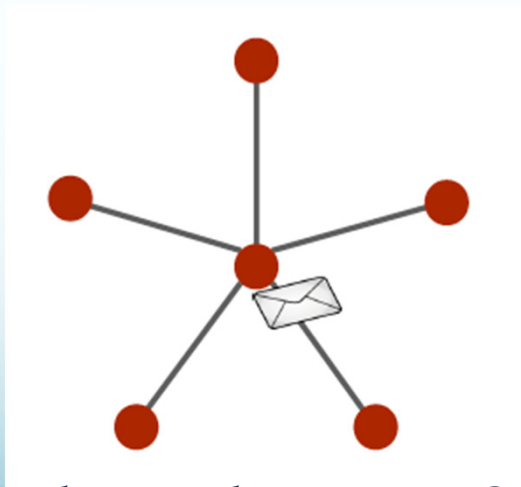
Comunicación: Gossiping

Protocolo Pull

- Visión simple del algoritmo desde el emisor
 1. $t=0$
 2. mientras que $t < T$ hacer
 3. cada agente desinformado llama a un vecino al azar, y obtiene el rumor si el vecino lo tiene.
 4. $t=t+1$

Comunicación: Gossiping

- PUSH
 - Los nodos con rumores envían a un vecino al azar
- PULL
 - Los nodos sin rumor le piden a un vecino al azar



Mal ejemplo para PUSH



Mal ejemplo para PULL

Comunicación: Gossiping

Protocolo PushPull

- Visión simple del algoritmo desde el emisor
 1. $t=0$
 2. mientras que $t < T$ hacer
 - 3.1 cada agente informado envía el rumor a su vecino aleatorio.
 - 3.2 cada agente desinformado llama a un vecino al azar, y obtiene el rumor si el vecino lo tiene.
 4. $t=t+1$

Comunicación: Gossiping

% Push version

on timeout

```
┌  $q \leftarrow \text{random}(P)$   
└ send  $\langle \text{PUSH}, \text{value} \rangle$  to  $q$   
└ set timeout  $\Delta$ 
```

on receive $\langle \text{PUSH}, v \rangle$

```
┌ if  $\text{value.time} < v.time$  then  
└  $\text{value} \leftarrow v$ 
```

% Pull version

on timeout

```
┌  $q \leftarrow \text{random}(P)$   
└ send  $\langle \text{PULL}, p, \text{value.time} \rangle$  to  $q$   
└ set timeout  $\Delta$ 
```

on receive $\langle \text{PULL}, q, t \rangle$

```
┌ if  $\text{value.time} > t$  then  
└ send  $\langle \text{REPLY}, \text{value} \rangle$  to  $q$ 
```

on receive $\langle \text{REPLY}, v \rangle$

```
┌ if  $\text{value.time} < v.time$  then  
└  $\text{value} \leftarrow v$ 
```

% Push-pull version

on timeout

```
┌  $q \leftarrow \text{random}(P)$   
└ send  $\langle \text{PUSHPULL}, p, \text{value} \rangle$  to  $q$   
└ set timeout  $\Delta$ 
```

on receive $\langle \text{PUSHPULL}, q, v \rangle$

```
┌ if  $\text{value.time} < v.time$  then  
└  $\text{value} \leftarrow v$   
else if  $\text{value.time} > v.time$  then  
└ send  $\langle \text{REPLY}, \text{value} \rangle$  to  $q$ 
```

on receive $\langle \text{REPLY}, v \rangle$

```
┌ if  $\text{value.time} < v.time$  then  
└  $\text{value} \leftarrow v$ 
```

Hay muchos más algoritmos

- Objetivo: asegurar la robustez y la eficiencia reduciendo complejidad

Práctica

- Implementar agentes en SPADE que incorporen los algoritmos Gossiping para compartir información
 - *Push* (os damos una posible implementación)
 - *Pull*
 - *Push-Pull*
- Un agente genera aleatoriamente un valor entero y lo empieza a difundir a k agentes (en principio $k=1$).
- Realizar pruebas donde **todos los agentes se quedan**:
 - **con el valor máximo (o el mínimo)**
 - **con la media**
- Comparar funcionamiento con 10, 20, 100 y 200 instancias de agentes
- Probar con $k>1$ (siendo k el numero de amigos a los que se manda mensaje en cada ronda)
- Comparar resultados (tiempo y número de mensajes enviados hasta converger)

Práctica

- A entregar (tarea Poliformat):
 - Código de cada versión en un fichero distinto .py (o en notebooks)
 - Documento con pequeño análisis de resultados
- Se puede hacer por parejas
- Fecha máxima: **17 de enero**