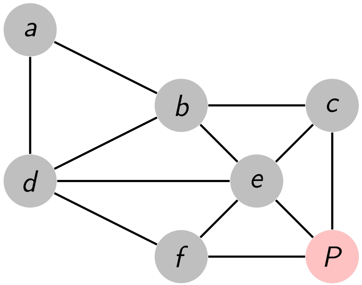
Routing Algorithms using Random Walks with Tabu Lists

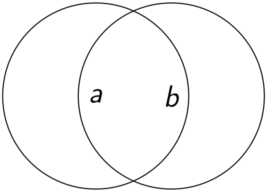
Karine Altisen, Stéphane Devismes, **Pascal Lafourcade** and Clément Ponsonnet

Verimag, University of Grenoble

ALGOTEL 2011 : May, Cap Estérel

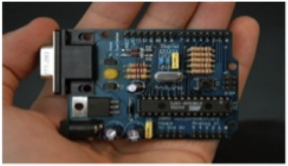
Wireless Sensor Network (WSN)



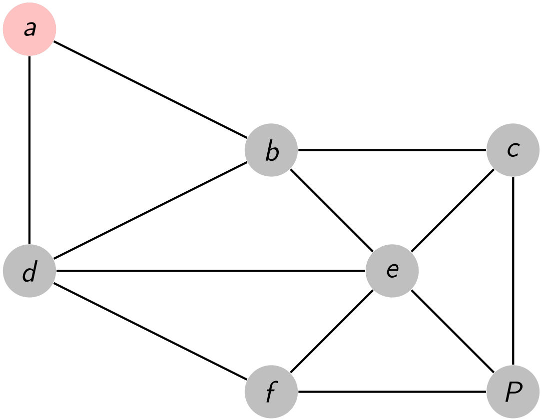


A node has :

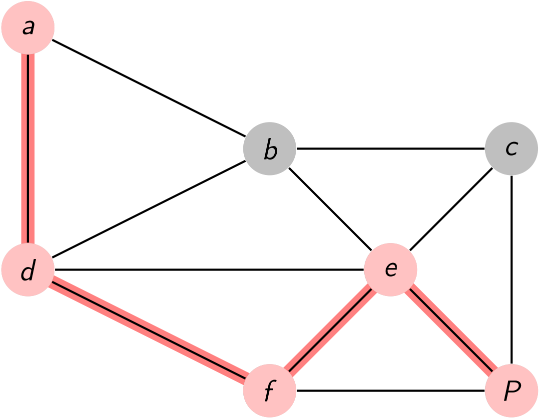
- low memory,
- low computation power
- finite battery.



Routing



Routing



Settings

- One sink/Multi source
- Connected
- Identified
- Reliable
- Asynchronous
- Spontaneous requests

Hitting Time

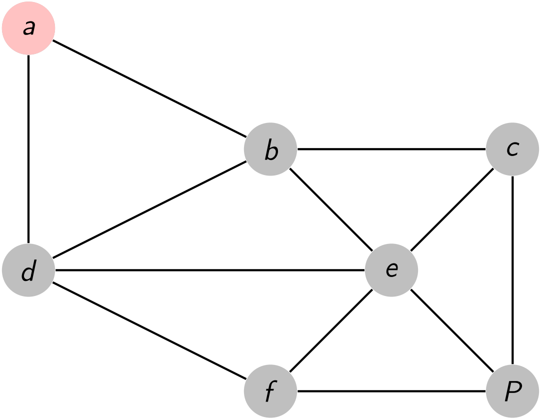
Average number of hops to reach the sink

Plan

Plan

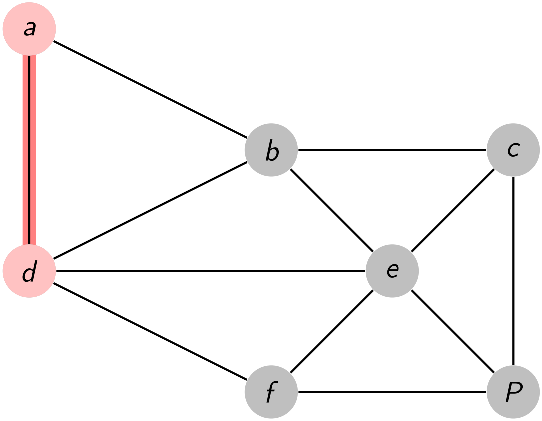
Random Walk

Uniform Law: $P(u) = \frac{1}{|\mathcal{N}_u|}$



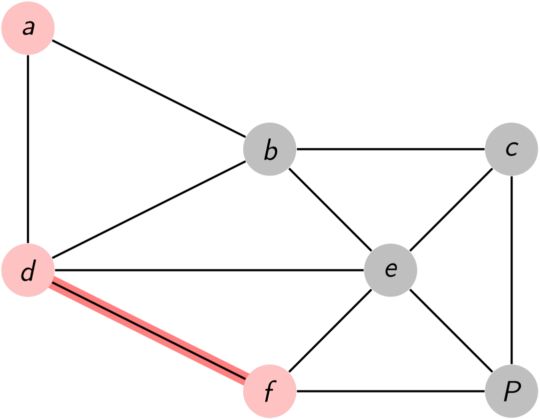
Random Walk

Uniform Law: $P(u) = \frac{1}{|\mathcal{N}_u|}$



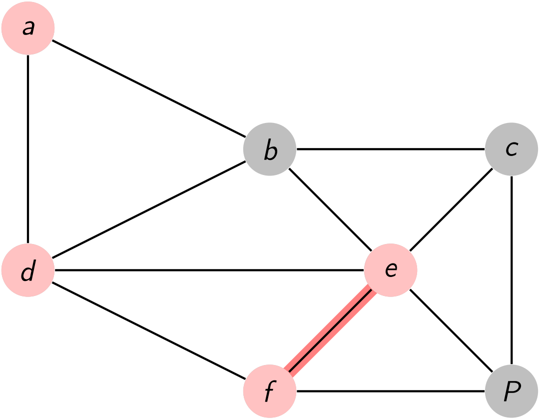
Random Walk

Uniform Law: $P(u) = \frac{1}{|\mathcal{N}_u|}$



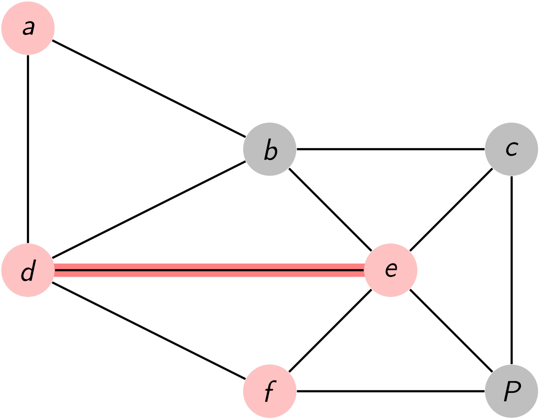
Random Walk

Uniform Law: $P(u) = \frac{1}{|\mathcal{N}_u|}$



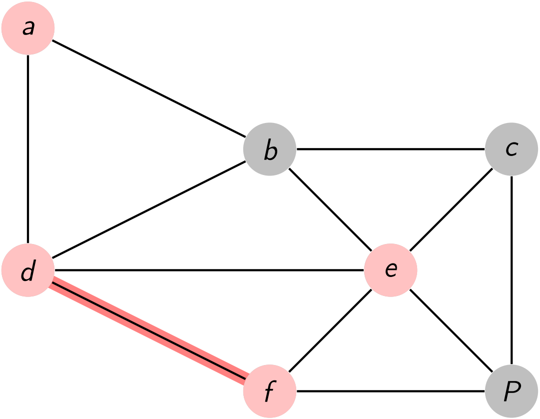
Random Walk

Uniform Law: $P(u) = \frac{1}{|\mathcal{N}_u|}$



Random Walk

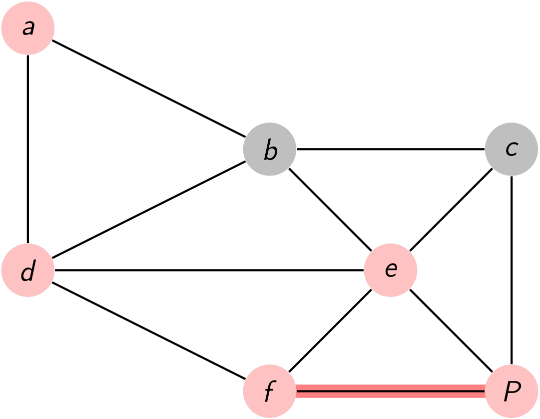
Uniform Law: $P(u) = \frac{1}{|\mathcal{N}_u|}$



Random Walk

Uniform Law: $P(u) = \frac{1}{|\mathcal{N}_u|}$

Complexity: $O(N^3)$



RWLD by Yamashita et al.

$\delta_u =$ number of neighbours of u

$$P(u) = \frac{\delta_u^{-\frac{1}{2}}}{\sum_{w \in \mathcal{N}_v} \delta_w^{-\frac{1}{2}}}$$

Complexity: $O(N^2)$

Avoiding high degree nodes

RWLD by Yamashita et al.

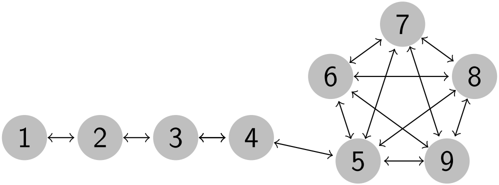
$\delta_u =$ number of neighbours of u

$$P(u) = \frac{\delta_u^{-\frac{1}{2}}}{\sum\limits_{w \in \mathcal{N}_v} \delta_w^{-\frac{1}{2}}}$$

Complexity: $O(N^2)$

Avoiding high degree nodes

Worst Case:



Routing by Random Walk

Pros

- Message length
- Tight local computation and memory
- No overlay
- No control messages (no load effect)
- ...

Cons

- Hitting time is $O(N^3)$ (RW) and $O(N^2)$ (RWLD)

Plan

Random Walk with Tabu Lists

- Idea : Add memory to help random walks.

  Avoiding cycles

- Store hints about previous choices

Is it a “good” trade-off ?

Where is the Tabu List?

Two Options

Where is the Tabu List?

Two Options

In Messages (Pessimistic)

In Nodes (Optimistic)

Same Goal: Detecting and Avoiding Cycles

How to deal with Full List?

Small memory and battery $\Rightarrow$ small finite size of tabu list.

- A node identity appears only once in the Tabu List.

How to deal with Full List?

Small memory and battery $\Rightarrow$ small finite size of tabu list.

- A node identity appears only once in the Tabu List.

Two Policies for updating.

How to deal with Full List?

Small memory and battery $\Rightarrow$ small finite size of tabu list.

- A node identity appears only once in the Tabu List.

Two Policies for updating.

Variation of FIFO (First In First out)

- If $N \in TL$ then N becomes first.
- Otherwise Add N and Remove the older node.

How to deal with Full List?

Small memory and battery ⇒ small finite size of tabu list.

- A node identity appears only once in the Tabu List.

Two Policies for updating.

Variation of FIFO (First In First out)

- If $N \in TL$ then N becomes first.
- Otherwise Add N and Remove the older node.

RAND

If $N \notin TL$ then

- Randomly remove a node in TL.
- Add N instead.

Otherwise keep the current TL.

Parameters of our algorithms

- 2 Locations of tabu lists: Message vs Nodes
- 2 Updates : FIFO vs RAND
- k = Size of tabu list

Plan

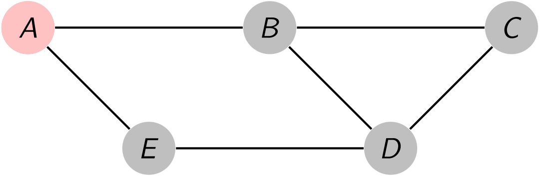
Idea of TLM

Avoiding Cycles!

- Store IDs of visited nodes.
- Visit new nodes first.

TLM(2,FIFOUpdate)

Size 2 avoids cycle of size 3.



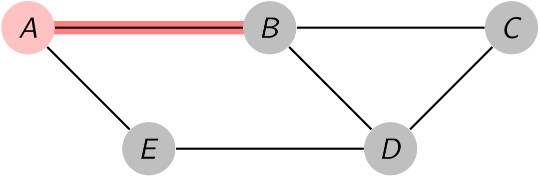
FIFO (First In First Out)

Visited Node : A

Tabu List:

TLM(2,FIFOUpdate)

Size 2 avoids cycle of size 3.



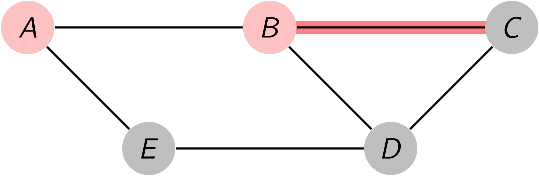
FIFO (First In First Out)

Visited Node : B

Tabu List:

TLM(2,FIFOUpdate)

Size 2 avoids cycle of size 3.



FIFO (First In First Out)

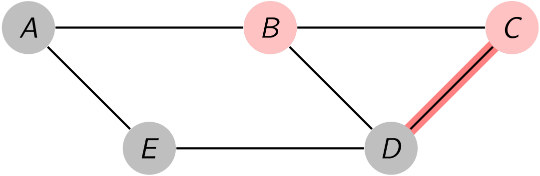
Visited Node : C

Tabu List:

A	B

TLM(2,FIFOUpdate)

Size 2 avoids cycle of size 3.



FIFO (First In First Out)

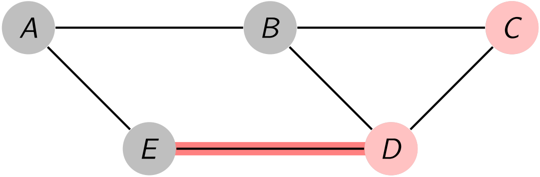
Visited Node : D

Tabu List:

B	C

TLM(2,FIFOUpdate)

Size 2 avoids cycle of size 3.



FIFO (First In First Out)

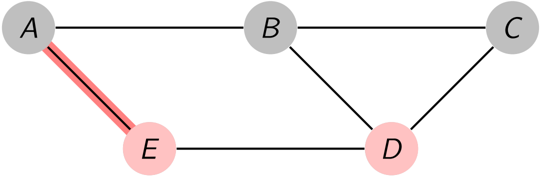
Visited Node : E

Tabu List:

C	D

TLM(2,FIFOUpdate)

Size 2 avoids cycle of size 3.



FIFO (First In First Out)

Visited Node : A

Tabu List: | D | E |

Summary of TLM

- Store in each message a list of already visited nodes.
- Select first nodes not in the Tabu List.
- Two updates policies : FIFO or RAND

Dead-end

If all neighbours are in the Tabu List
then we use RWLD.

Plan

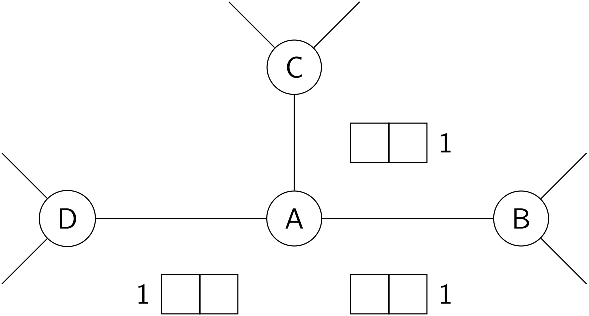
Idea of TLCN

Detect cycles by learning.
Avoid to route to cycles.

- One list per destination : Store message ID.
- One counter per destination : Detection of cycles.

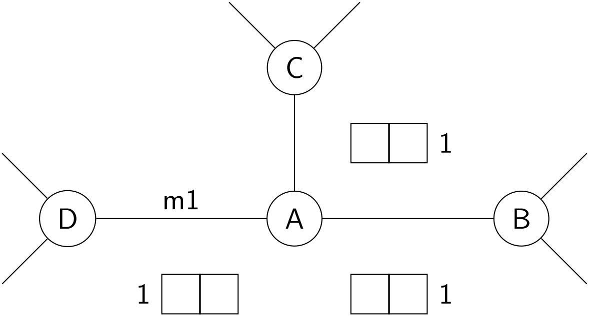
TLCN(2,FIFOUpdate)

Cycle detection



TLCN(2,FIFOUpdate)

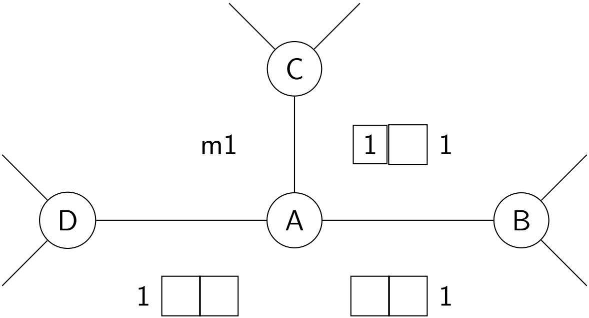
Cycle detection



$$P^{v}_{\mathrm{TLCN}}(Counter)(u) = \frac{Counter[u]^{-1} \times \delta_u^{-1/2}}{\sum_{w \in \mathcal{N}_v} Counter[w]^{-1} \times \delta_w^{-1/2}}$$

TLCN(2,FIFOUpdate)

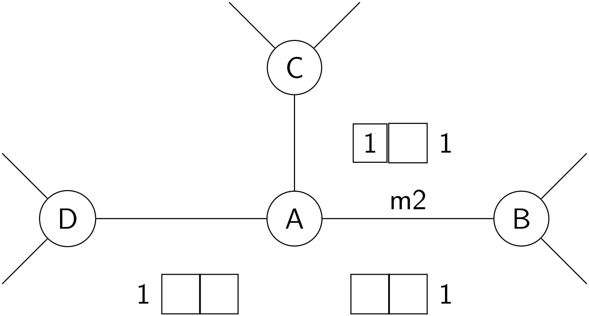
Cycle detection



$$P^{v}_{\mathrm{TLCN}}(Counter)(u) = \frac{Counter[u]^{-1} \times \delta_u^{-1/2}}{\sum_{w \in \mathcal{N}_v} Counter[w]^{-1} \times \delta_w^{-1/2}}$$

TLCN(2,FIFOUpdate)

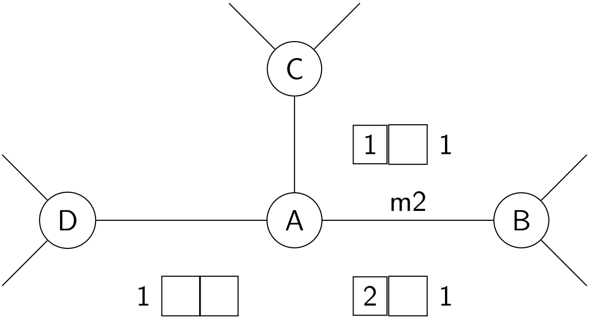
Cycle detection



$$P^{v}_{\mathrm{TLCN}}(Counter)(u) = \frac{Counter[u]^{-1} \times \delta_u^{-1/2}}{\sum_{w \in \mathcal{N}_v} Counter[w]^{-1} \times \delta_w^{-1/2}}$$

TLCN(2,FIFOUpdate)

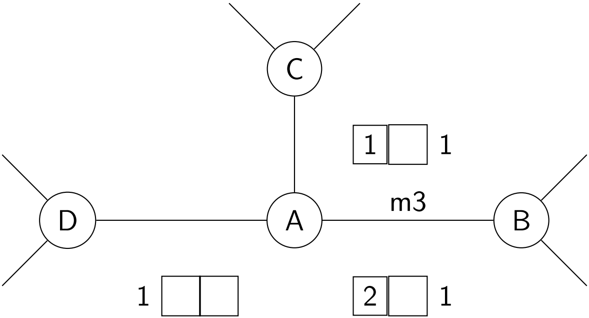
Cycle detection



$$P^{v}_{\mathrm{TLCN}}(Counter)(u) = \frac{Counter[u]^{-1} \times \delta_u^{-1/2}}{\sum_{w \in \mathcal{N}_v} Counter[w]^{-1} \times \delta_w^{-1/2}}$$

TLCN(2,FIFOUpdate)

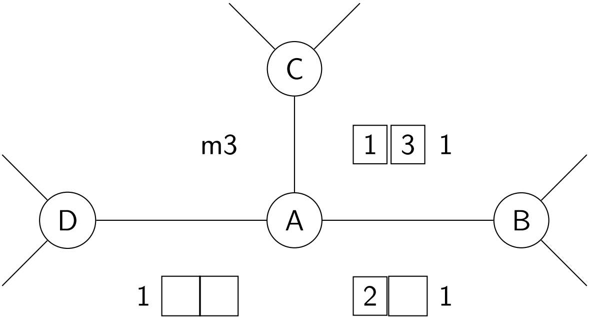
Cycle detection



$$P^v_{\mathrm{TLCN}}(Counter)(u) = \frac{Counter[u]^{-1} \times \delta_u^{-1/2}}{\sum_{w \in \mathcal{N}_v} Counter[w]^{-1} \times \delta_w^{-1/2}}$$

TLCN(2,FIFOUpdate)

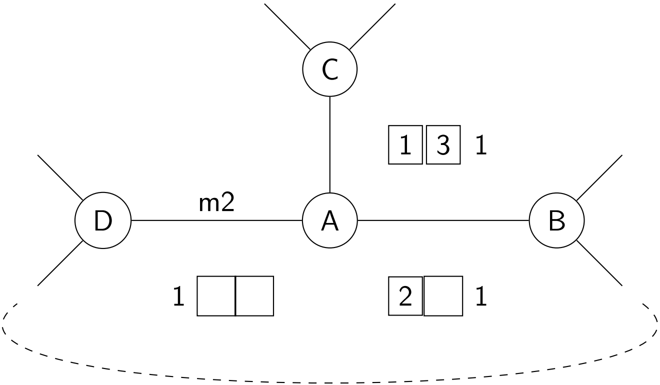
Cycle detection



$$P^{v}_{\mathrm{TLCN}}(Counter)(u) = \frac{Counter[u]^{-1} \times \delta_u^{-1/2}}{\sum_{w \in \mathcal{N}_v} Counter[w]^{-1} \times \delta_w^{-1/2}}$$

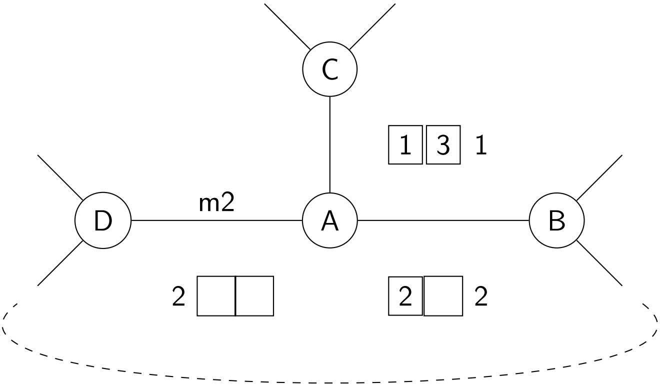
TLCN(2,FIFOUpdate)

Cycle detection



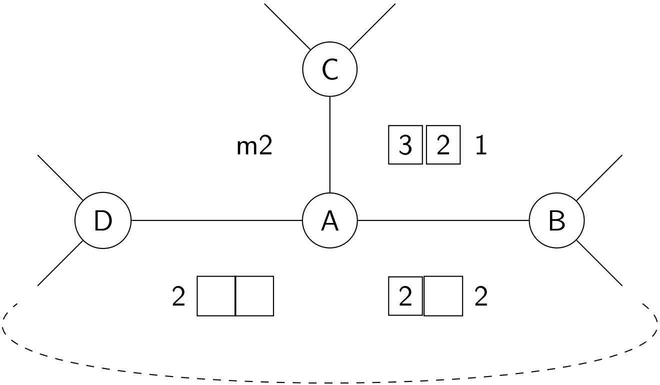
TLCN(2,FIFOUpdate)

Cycle detection



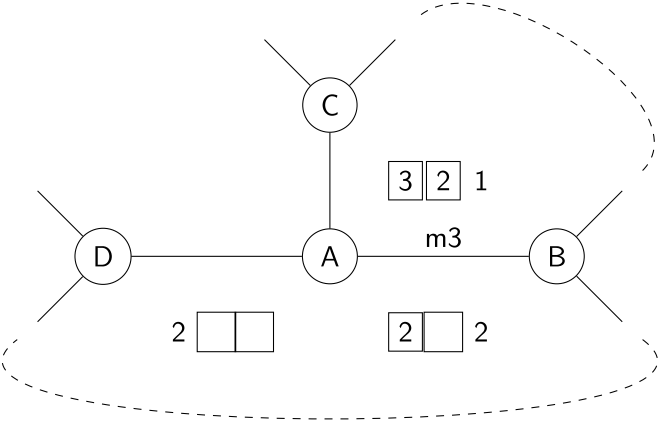
TLCN(2,FIFOUpdate)

Cycle detection



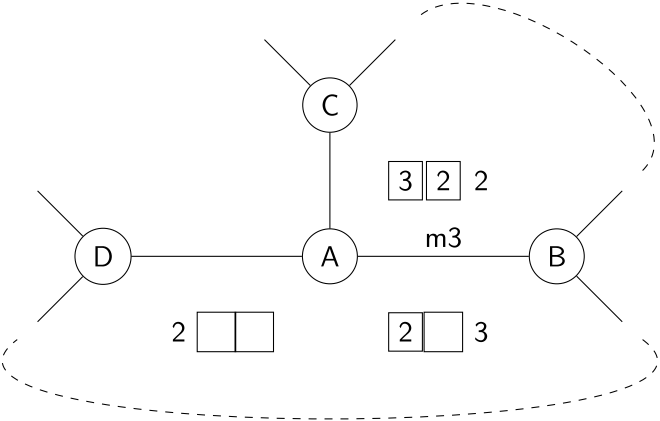
TLCN(2,FIFOUpdate)

Cycle detection



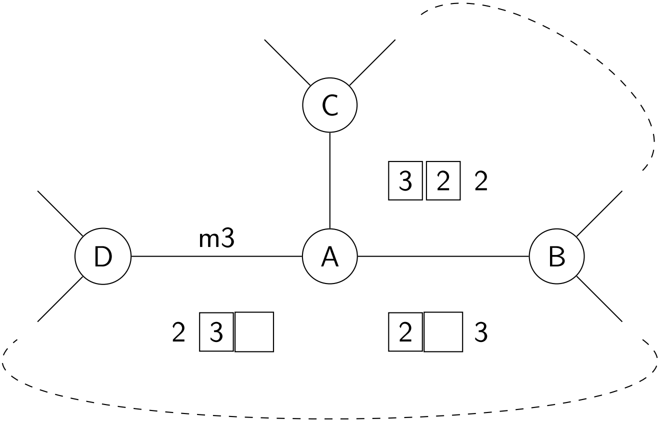
TLCN(2,FIFOUpdate)

Cycle detection



TLCN(2,FIFOUpdate)

Cycle detection



Summary of TLCN

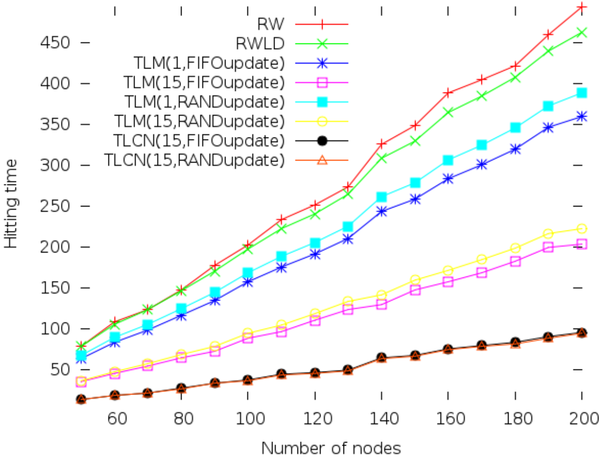
- In each node for each link :
  - Tabu List of messages already send.
  - Counter detecting possible cycle.
- Prefer to visit link with a low counter.
- Two updates policies : FIFO or RAND.

Plan

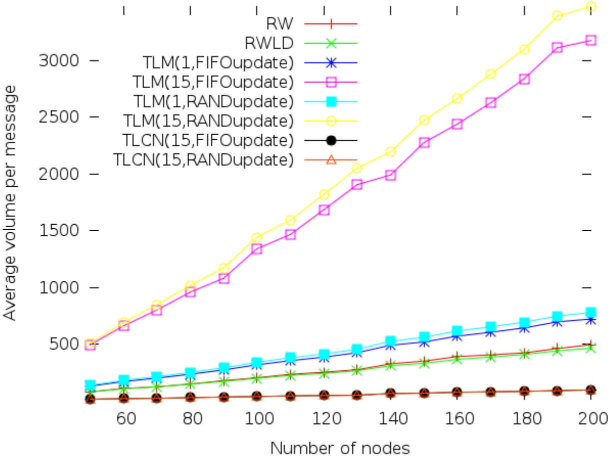
Experiment Settings

- Sinalgo(JAVA)
- Graphs:
  - UDG.
  - connected.
  - one sink/multi-source.
  - uniform distribution.
  - Size : 60, 70, . . . , 200 nodes.
- 100 messages per sources.

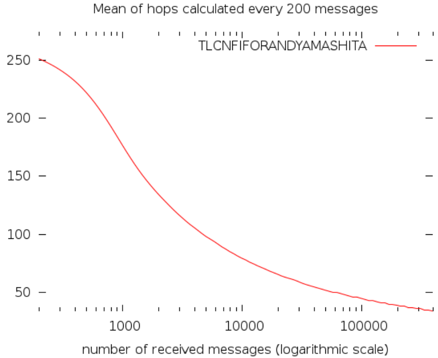
Hitting Time and Number of Nodes



Volume and Number of Nodes



Convergence of TLCN



If size(TL)=∞ then TLCN converges to spanning tree.

Plan

Random Walk with Tabu List

TLM vs RW

- Better Hitting time.
- Introduces an overload.
- Size of the list has an influence.
- FIFO is slightly better than RAND.

TLCN vs RW

- Improve Random Walk.
- Size of the list has a small influence.
- FIFO = RAND

Conclusion : TLCN better than TLM better than RW

Sum Up

	Hitting Time	Volume	Load Sensitivity
TLCN (15,FIFO)	1^{st}	1^{st}	yes
TLCN (15,RAND)	1^{st}	1^{st}	yes
TLM (15,FIFO)	3^{rd}	7^{th}	no
TLM (15,RAND)	4^{th}	8^{th}	no
TLM (1,FIFO)	5^{th}	5^{th}	no
TLM (1,RAND)	6^{th}	6^{th}	no
RWLD	7^{th}	3^{rd}	no
RW	8^{th}	4^{th}	no

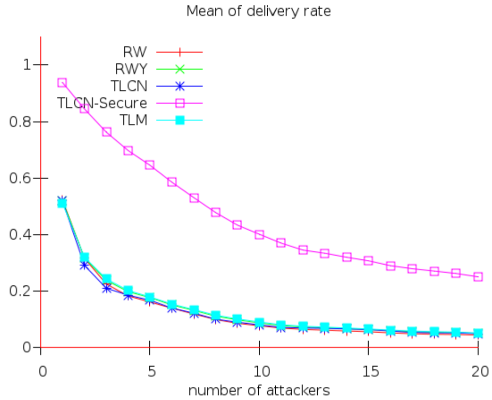
Next

- Theoretical study of TLM and TLCN :
  - for any graph, TLM(1,FIFO) $>$ RW,
  - Find graph such that $k > 2$
    TLM(k, FIFO) <> TLM (k+1,FIFO).
  - Optimal size for Tabu List in TLCN, TLM.
  - Convergence of TLCN.
- Resistance to attackers?
- How to secure our protocols?

Thanks for your Attention

Questions ?

Results for Secured Version of TLCN



Results for Secured Version of TLCN