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.
Settings

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

Hitting Time
Average number of hops to reach the sink
Plan

Introduction
Random Walks
Random Walks with Tabu Lists
In Messages
In Nodes
Experimental Results
Conclusion
Plan

Introduction

Random Walks

Random Walks with Tabu Lists

In Messages

In Nodes

Experimental Results

Conclusion
Random Walk

Uniform Law: $P(u) = \frac{1}{|N_u|}$
Random Walk

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

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

Uniform Law: \( P(u) = \frac{1}{|N_u|} \)
Random Walk

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

Uniform Law: \( P(u) = \frac{1}{|N_u|} \)

Complexity: \( O(N^3) \)
RWLD by Yamashita et al.

\[ \delta_u = \text{number of neighbours of } u \]

\[ P(u) = \frac{\delta_u^{-\frac{1}{2}}}{\sum_{w \in N_v} \delta_w^{-\frac{1}{2}}} \]

Complexity: \( O(N^2) \)

Avoiding high degree nodes
RWLD by Yamashita et al.

\[ \delta_u = \text{number of neighbours of } u \]

\[ P(u) = \frac{\delta_u^{-\frac{1}{2}}}{\sum_{w \in N_v} \delta_w^{-\frac{1}{2}}} \]

Avoiding high degree nodes

Complexity: \( O(N^2) \)

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

Introduction

Random Walks

Random Walks with Tabu Lists

In Messages

In Nodes

Experimental Results

Conclusion
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 ⇒ 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 $\Rightarrow$ 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.

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

Introduction
Random Walks
Random Walks with Tabu Lists
In Messages
In Nodes
Experimental Results
Conclusion
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:  

A
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

Introduction

Random Walks

Random Walks with Tabu Lists

In Messages

In Nodes

Experimental Results

Conclusion
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

![Diagram of a network with nodes A, B, C, and D connected by edges with numbers 1]
TLCN(2, FIFOUpdate)

Cycle detection

\[ P_{\text{TLCN}}(\text{Counter})(u) = \frac{\text{Counter}[u]^{-1} \times \delta_u^{-1/2}}{\sum_{w \in N_v} \text{Counter}[w]^{-1} \times \delta_w^{-1/2}} \]
TLCN(2,FIFOUpdate)

Cycle detection

\[ P_{\text{TLCN}}^{v}(\text{Counter})(u) = \frac{\text{Counter}[u]^{-1} \times \delta_u^{-1/2}}{\sum_{w \in N_v} \text{Counter}[w]^{-1} \times \delta_w^{-1/2}} \]
TLCN(2, FIFOUpdate)

Cycle detection

\[ P_{\text{TLCN}}(\text{Counter})(u) = \frac{\text{Counter}[u]^{-1} \times \delta_u^{-1/2}}{\sum_{w \in \mathcal{N}_v} \text{Counter}[w]^{-1} \times \delta_w^{-1/2}} \]
**TLCN(2,FIFOUpdate)**

Cycle detection

![Diagram showing nodes A, B, C, and D with counters and transitions labeled with 1 and 2.]

\[
P_{TLCN}^v(\text{Counter})(u) = \frac{\text{Counter}[u]^{-1} \times \delta_u^{-1/2}}{\sum_{w \in \mathcal{N}_v} \text{Counter}[w]^{-1} \times \delta_w^{-1/2}}
\]
TLCN(2,FIFOUpdate)

Cycle detection

\[ P_{\text{TLCN}}^v(\text{Counter})(u) = \frac{\text{Counter}[u]^{-1} \times \delta_u^{-1/2}}{\sum_{w \in N_v} \text{Counter}[w]^{-1} \times \delta_w^{-1/2}} \]
TLCN(2,FIFOUpdate)

Cycle detection

\[
P_{\text{TLCN}}(\text{Counter})(u) = \frac{\text{Counter}[u]^{-1} \times \delta_u^{-1/2}}{\sum_{w \in \mathcal{N}_v} \text{Counter}[w]^{-1} \times \delta_w^{-1/2}}
\]
TLCN(2,FIFOUpdate)

Cycle detection

Graph: A connected to B, C, and D. Edges labeled with numbers: m2, 1, 3, 1, 2, 1.
TLCN(2,FIFOUpdate)

Cycle detection

1 3 1
m2
2
2 2

A
B
TLCN(2,FIFOUpdate)

Cycle detection

![Diagram with nodes A, B, C, D and edges labeled m2, 3, 2, 1, 2, 2, 2, and a dashed line indicating a cycle]
TLCN(2,FIFOUpdate)

Cycle detection

A

B

C

D

m3

2

2

3 2 1

2 2 2
TLCN(2,FIFOUpdate)

Cycle detection

Diagram showing nodes A, B, C, and D with edges and numbers indicating movement and cycles.
TLCN(2,FIFOUpdate)

Cycle detection

D → m3 → A → B

A → C

3 2 2

2 3

2 3
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

Introduction
Random Walks
Random Walks with Tabu Lists
In Messages
In Nodes
Experimental Results
Conclusion
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

The graph shows the average volume per message as a function of the number of nodes. Different lines represent various methods:

- RW
- RWLD
- TLM(1,FIFOupdate)
- TLM(15,FIFOupdate)
- TLM(1,RANDupdate)
- TLM(15,RANDupdate)
- TLCN(15,FIFOupdate)
- TLCN(15,RANDupdate)

The x-axis represents the number of nodes, ranging from 60 to 200, and the y-axis represents the average volume per message, ranging from 500 to 3000.
Convergence of TLCN

If \( \text{size}(\text{TL}) = \infty \) then TLCN converges to spanning tree.
Plan

Introduction
Random Walks
Random Walks with Tabu Lists
In Messages
In Nodes
Experimental Results
Conclusion
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

<table>
<thead>
<tr>
<th></th>
<th>Hitting Time</th>
<th>Volume</th>
<th>Load Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLCN (15,FIFO)</td>
<td>1st</td>
<td>1st</td>
<td>yes</td>
</tr>
<tr>
<td>TLCN (15,RAND)</td>
<td>1st</td>
<td>1st</td>
<td>yes</td>
</tr>
<tr>
<td>TLM (15,FIFO)</td>
<td>3rd</td>
<td>7th</td>
<td>no</td>
</tr>
<tr>
<td>TLM (15,RAND)</td>
<td>4th</td>
<td>8th</td>
<td>no</td>
</tr>
<tr>
<td>TLM (1,FIFO)</td>
<td>5th</td>
<td>5th</td>
<td>no</td>
</tr>
<tr>
<td>TLM (1,RAND)</td>
<td>6th</td>
<td>6th</td>
<td>no</td>
</tr>
<tr>
<td>RWLD</td>
<td>7th</td>
<td>3rd</td>
<td>no</td>
</tr>
<tr>
<td>RW</td>
<td>8th</td>
<td>4th</td>
<td>no</td>
</tr>
</tbody>
</table>
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

Mean of delivery rate

- RW
- RWY
- TLCN
- TLCN-Secure
- TLM

number of attackers

0 5 10 15 20

0 0.2 0.4 0.6 0.8 1
Results for Secured Version of TLCN

Mean of delivery rates to each receipt of 200 messages

Number of received messages (logarithmic scale)

TLCNSECUREFIFOYAMYAM