Analysis of Link Reversal Routing Algorithms for Mobile Ad Hoc Networks

Costas Busch (RPI)
Srikanth Surapaneni (RPI)
Srikanta Tirthapura (Iowa State University)
Talk Outline

Link Reversal Routing
Previous Work & Contributions
Analysis of Full Reversal Algorithm
Analysis of Partial Reversal Algorithm
Analysis of Deterministic Algorithms
Conclusions
Link Reversal Routing

Connection graph of a mobile network

Destination oriented, acyclic graph

Destination
Link Failure

node moves
- **Bad node:** no path to destination

- **Good node:** at least one path to destination
**Full** Link Reversal Algorithm

Sinks reverse **all** their links

#reversals = 7      time = 5
Partial Link Reversal Algorithm

Sinks reverse *some* of their links

#reversals = 5       time = 5
Heights

**General height:** $(a_1, a_2, \ldots, a_n)$

Heights are ordered in lexicographic order.
**Full Link Reversal Algorithm**

Node $i$: $(a_i, i)$

- **Real height**
- **Node ID** (breaks ties)
**Full Link Reversal Algorithm**

**Sink** $i$:

before reversal

$(a_i, i)$

after reversal

$(a_i', i)$

$$a_i' = \max\{a_j : j \in N(i)\} + 1$$
Full Link Reversal Algorithm
Partial Link Reversal Algorithm

Node $i$: $(a_i, b_i, i)$

- Real height
- memory
- Node ID (breaks ties)
Partial Link Reversal Algorithm

Sink \( i \): before reversal \((a_i, b_i, i)\) after reversal \((a'_i, b'_i, i)\)

\[ a'_i = \min\{ a_j : j \in N(i) \} + 1 \]

\[ b'_i = \min\{ b_j : j \in N(i) \text{ and } a'_i = b'_j \} - 1 \]
**Partial Link Reversal Algorithm**
Deterministic Link Reversal Algorithms

Sink $i$:

Before reversal:

$\mathbf{h}$

After reversal:

$h' = g(h, h_1, h_2, ..., h_k)$

Deterministic function
Interesting measures:

#reversals: total number of node reversals (work)

Time: time needed to reach a good state (stabilization time)
Talk Outline

Link Reversal Routing

Previous Work & Contributions
Analysis of Full Reversal Algorithm
Analysis of Partial Reversal Algorithm
Analysis of Deterministic Algorithms
Conclusions
Previous Work

• Introduction of the problem
• First proof of stability

• LMR – Lightweight Mobile Routing Alg.

Park and Corson: INFOCOM 1997
• TORA – Temporally Ordered Routing Alg.
  Variation of partial reversal
  Deals with partitions
Previous Work

  • Leader election based on TORA
  • (partial) proof of stability

Experimental work and surveys:

Broch et al.: MOBICOM 1998
Samir et al.: IC3N 1998
Rajamaran: SIGACT news 2002
Contributions

First formal performance analysis of link reversal routing algorithms in terms of #reversals and time
Contributions

Full reversal algorithm:

#reversals and time: $O(n^2)$
There are worst-cases with: $\Omega(n^2)$

Partial reversal algorithm:

#reversals and time: $O(n \cdot a^* + n^2)$
There are worst-cases with: $\Omega(n \cdot a^* + n^2)$

$a^*$ depends on the network state
Contributions

Any deterministic algorithm:

There are states such that

\# reversals and time: \( \Omega(n^2) \)

\[ n \text{ bad nodes} \]

Full reversal is worst-case optimal \( \Omega(n^2) \)

Partial reversal is not! \( \Omega(n \cdot a^* + n^2) \)
Link Reversal Routing
Previous Work & Contributions
Analysis of Full Reversal Algorithm
Analysis of Partial Reversal Algorithm
Analysis of Deterministic Algorithms
Conclusions
For any execution of the full reversal algorithm:

- \#reversals is the same
- Final state is the same

(this holds for any deterministic algorithm)
Bad state

dest.

Good nodes

Bad nodes
Layers of bad nodes

dest.

Good nodes  Bad nodes

$L_1$  $L_2$  $L_3$  $L_4$
Layers of bad nodes

A layer:

dest.
There is an execution $E_1$ such that:
Every bad node reverses exactly once
There is an execution $E_1$ such that:
Every bad node reverses exactly once
There is an execution $E_1$ such that:
Every bad node reverses exactly once
There is an execution $E_1$ such that:

Every bad node reverses exactly once
At the end of execution $E_1$:

- All nodes of layer $L_1$ become good nodes
- The remaining bad nodes return to the same state as before the execution
At the end of execution $E_1$:

- All nodes of layer $L_1$ become good nodes

- The remaining bad nodes return to the same state as before the execution
There is an execution $E_2$ such that:
Every bad node reverses exactly once
At the end of execution $E_2$:

- All nodes of layer $L_2$ become good nodes

- The remaining bad nodes return to the same state as before the execution
At the end of execution $E_2$:

- All nodes of layer $L_2$ become good nodes
- The remaining bad nodes return to the same state as before the execution
At the end of execution $E_3$:
All nodes of layer $L_3$ become good nodes
At the end of execution $E_4$:

All nodes of layer $L_4$ become good nodes
Reversals per node: 0 0 0 0 ... 0

dest.
Reversals per node: 1 1 1 1 ... 1

End of execution $E_1$
Reversals per node:  1  2  2  \ldots  2

End of execution $E_2$
Reversals per node: 1 2 3 ... 3

End of execution $E_3$
Reversals per node: \(1 \ 2 \ 3 \ \ldots \ m\)

End of execution \(E_m\)
Reversals per node: 1, 2, 3, ... , m

Each node in layer $L_i$ reverses $i$ times
Reversals per node: \(1 \quad 2 \quad 3 \quad \ldots \quad m\)

Nodes per layer: \(n_1 \quad n_2 \quad n_3 \quad \ldots \quad n_m\)

\#reversals: \(1 \cdot n_1 + 2 \cdot n_2 + 3 \cdot n_3 + \ldots + m \cdot n_m\)
For $n$ bad nodes, trivial upper bound: $O(n^2)$

(\#reversals and time)

\#reversals: $1 \cdot n_1 + 2 \cdot n_2 + 3 \cdot n_3 + \cdots + m \cdot n_m$

$n \cdot n$
\( O(n^2) \) \#reversals bound is tight

Reversals per node:

\[
\text{\#reversals: } 1 + 2 + 3 + \cdots + n = \Omega(n^2)
\]
None of these reversals are performed in parallel.

\[ \text{None of these reversals are performed in parallel} \]

\[ \frac{n}{2} + 1 = \Omega(n^2) \]

\[ \text{Time needed} \quad \Omega(n^2) \]
Talk Outline

Link Reversal Routing
Previous Work & Contributions
Analysis of Full Reversal Algorithm
Analysis of Partial Reversal Algorithm
Analysis of Deterministic Algorithms
Conclusions
Layers of bad nodes

Nodes at layer $L_i$ are at distance $i$ from good nodes

Good nodes  Bad nodes

dest.
Layers of bad nodes

(dest.)

\( (a_i, b_i, i) \)

\( (a^\text{max}, b_i, i) \)

\( (a^\text{min}, b_i, i) \)

alpha value
when the network reaches a good state:

upper bound on alpha value
when the network reaches a good state:

upper bound on reversals per node

$$a^* = a_{\text{max}} - a_{\text{min}}$$
when the network reaches a good state:

For $n$ bad nodes: a bad node reverses at most $a^* + n$ times

#reversals and time: $O(n \cdot a^* + n^2)$
Reversals bound is tight

$O(n \cdot a^* + n^2)$

Reversals per node:

$\begin{align*}
\text{dest.} & \rightarrow L_1 \rightarrow L_2 \rightarrow L_3 \rightarrow L_n \\
& \quad a^* + 1 \quad + 2 \quad + 3 \quad + n
\end{align*}$

# reversals: $\Omega(n \cdot a^* + n^2)$
None of these reversals are performed in parallel.

\[ O(n \cdot a^* + n^2) \] time bound is tight.

\[ \text{#nodes} = \frac{n}{2} \]

\[ \Omega(n \cdot a^* + n^2) \]

\[ \text{Time needed} = \Omega(n \cdot a^* + n^2) \]
Talk Outline

Link Reversal Routing
Previous Work & Contributions
Analysis of Full Reversal Algorithm
Analysis of Partial Reversal Algorithm
Analysis of Deterministic Algorithms
Conclusions
Layers of bad nodes

Nodes at layer $L_i$ are at distance $i$ from good nodes.
Layers of bad nodes

for any height function $g$, there is an initial assignment of heights such that......
when the network reaches a good state:

lower bound on reversals per node
Lower Bound on \#reversals

Reversals per node:

\[ \text{dest.} \xrightarrow{0} L_1 \xrightarrow{1} L_2 \xrightarrow{2} L_3 \xrightarrow{n-1} L_n \]

\#reversals: \[ 1 + 2 + 3 + \cdots + n = \Omega(n^2) \]
None of these reversals are performed in parallel.

Lower Bound on time

\#reversals in layer $L_{n/2}$:

\[
\frac{n(n - 1)}{2}\]

\[\Omega(n^2)\]

\#nodes = \[
\frac{n}{2}
\]

Time needed \[\Omega(n^2)\]
Talk Outline

Link Reversal Routing
Previous Work & Contributions
Analysis of Full Reversal Algorithm
Analysis of Partial Reversal Algorithm
Analysis of Deterministic Algorithms

Conclusions
• We gave the first formal performance analysis of deterministic link reversal algorithms

Open problems:

• Improve worst-case performance of partial link reversal algorithm

• Analyze randomized algorithms

• Analyze average-case performance