Fair Election of Monitoring Nodes in WSNs

Quentin Monnet[†] Youcef Hammal[‡] Lynda Mokdad[†]
Jalel Ben-Othman[*]

[†]LACL
Université Paris-Est (FR)

[‡]LSI
USTHB (DZ)

[*]L2TI
Université Paris 13 (FR)

Speaker: Pr Lynda Mokdad <lynda.mokdad@u-pec.fr>

Outline

1. Context
   - WSNs and security
   - Detection model
2. Proposal
   - Previous schemes
   - Selection principle
   - Selection algorithm
3. Simulation and Conclusion
   - Simulation results
   - Conclusion and future work

Outline

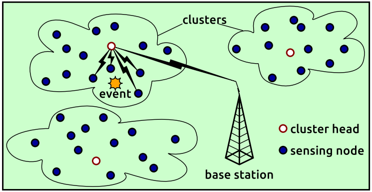
Wireless Sensor Networks (WSNs)

Small devices

- realize measurements (sensors)
- may be grouped into clusters
- ad-hoc communication
- linked to a base station (BS)
- civil and military applications

Restricted resources

- few computation capabilities
- few memory available
- few energy available (battery)



Security in WSNs

Many security issues in WSNs:

- Confidentiality (privacy of transmitted data)
- Authentication, integrity (proving identity, ensuring unaltered messages)
- Reliability, availability (ensuring network/services run as expected, whatever happens)
  Focus on: detecting denial of service attacks (DoS).
  Several layers for attacks:
  - physical layer (jamming *et cætera*)
  - MAC layer (jamming, greedy attacks, sleep deprivation, *et cætera*)
  - routing layer (black/sink/worm holes, routing attacks *et cætera*)
  - transport layer (TCP/UDP SYN/ACK flood *et cætera*)
  - application layer

Attack model

We want to detect compromised nodes trying to harm the network from the inside.

- The compromised sensor is part of the network
- It performs attacks on MAC or routing layers, *e.g.* flooding or black hole attacks

Detection model

We use a solution based on "watchdogs": control nodes (*cNodes*) which monitor their neighbors.

- *cNodes* apply rules to detect compromised nodes, *e.g.*:
  - data emission rate or emitted packets number of a neighbor should not exceed given threshold (jamming detection)
  - all packets sent to a neighbor for forwarding must be actually forwarded (black hole detection)
- On rule infringement, the suspicious sensor is reported to the cluster head
- On reception of several reports, cluster head virtually excludes suspicious node from the cluster.

Outline

1. Context
   - WSNs and security
   - Detection model

2. Proposal
   - Previous schemes
   - Selection principle
   - Selection algorithm

3. Simulation and Conclusion
   - Simulation results
   - Conclusion and future work

How to select *cNodes*?

We focus on the *cNodes* selection algorithm.

To ensure an efficient use of *cNode* role on must keep in mind:

- All nodes must be monitored (not necessarily at the same time)
- It consumes more energy than normal sensing

This led to proposal (in previous work) of a periodic renewal of the *cNodes* set.

- $\rightarrow$ Better load repartition between all sensors
- $\rightarrow$ Monitored area is virtually extended

Random selection

There are several ways to select *cNodes*.

In previous work each sensor could self-elect itself as a *cNode* in a pseudo-random manner (self-election similar to LEACH cluster-head selection process).

- In this way each node becomes *cNode* sooner or later
- Hence all nodes are monitored (as long as they have neighbors), even if not for all cycles

But energy consumption could be better balanced.

Energy-based selection

Another idea: sensors with the highest residual energy are selected.

At the end of each period, all nodes send the value of their residual energy to the cluster head, which designates the n requested *cNodes* for the new cycle.

But there are some issues:

- relative to security (nodes can “lie” about their residual energy to force the CH to choose them)
- relative to cluster coverage (the n nodes with the highest residual energy could always be in the same area of the cluster)

Our solution: democratic *cNodes* election

Proposed scheme

Reuse observations from the *cNodes*;
Have the outgoing *cNodes* vote for the new ones ("democratic" vote)

Democratic election — initialization

The democratic election scheme is an iterative process.

But at first there are no previous *cNodes* to rely on. We need an initialization phase:

- Each node i of the cluster sends to the CH the value of its residual energy, which is stored into a related array $RE_0[i]$
- Each node acts as a *cNode* and starts controlling its neighbors, and keeps transmitting data at the same time

Democratic election — main loop

1. At the end of iteration k, the CH asks each node i to send its residual energy value $RE_k[i]$ and asks each *cNode* j to send the array $Obs_k[j]$ containing its observations over transmission rates of its neighbors
2. For each node i, the cluster head acts as follows:
   - it assesses the observed energy consumption $ECa = \max(\{Obs_k[j][i]\}|_{j \in \mathcal{CN}})$ (max rate observed by *cNodes*)
   - it computes the declared energy consumption $ECd = RE_{k-1}[i] - RE_k[i]$ (difference of residual energy between the last two steps)
   - if $|ECd - ECa| \leq \epsilon$ then node i is declared as sane, and added to the set of eligible nodes; else $SSN[i]$ is increased by 1
   - if $SSN[i] \geq$ *threshold* then the node is declared as compromised
3. The CH selects *cNodes* from the set of eligible nodes (which prevents the security issue) in such a way that every node is controlled by at least two *cNodes* (which prevents coverage issue)
4. The data transmission period begins; at its end, loop to step #1

Democratic election — Main loop, Step #3: selection

At step #3 of main loop the CH selects the *cNodes* amongst the set of eligible nodes.
This selection is based on the votes of the nodes of the cluster. The CH designates the sensors with the highest residual energy and with correct (sane) behavior.

But depending on the application running in the network, it could also consider other relevant criteria, such as:

- connectivity index (number of direct neighbors)
- signal power
- *et cætera*

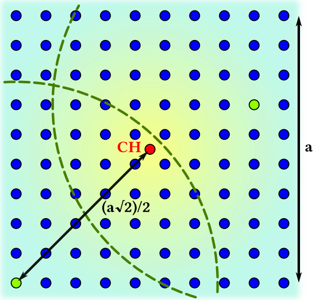
Outline

1. Context
   - WSNs and security
   - Detection model
2. Proposal
   - Previous schemes
   - Selection principle
   - Selection algorithm
3. Simulation and Conclusion
   - Simulation results
   - Conclusion and future work

Simulation parameters

Comparison between random selection and democratic election of *cNodes* (ns-2)

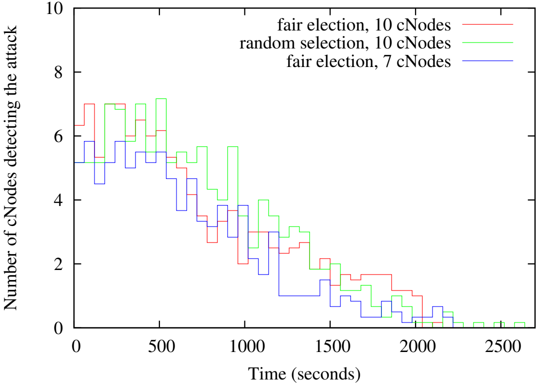
PARAMETER	VALUE
Simulation time	3,600 seconds
Initialization phase	30 seconds
Number of sensors	100 (+ cluster head)
Topology	regular grid (72m$\times$72m)
Compromised node	1
cNodes percentage	7 to 10 %
Transmission rate	1 kbits/s — 35 kbits/s*
Transmission range	50 meters
Packets size	500 bytes
Reception consumption	0.395 W
Emission consumption	0.660 W
Initial energy amount	10 J — ∞*



(*) value used only for compromised node

Attack detection

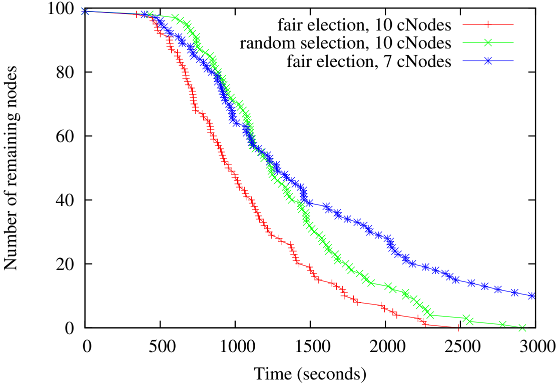
The efficiency is nearly equivalent between random and democratic election. It is more efficient with the democratic method after 1500 seconds as it selects a fixed amount of *cNodes* (versus a percentage of remaining nodes for random selection). Lowering the number of *cNodes* still leads to a good detection rate.



Network lifetime

The initialization phase consumes more (all nodes act as *cNodes* and have a high consumption).

The nodes die faster with the democratic election, but its efficiency regarding detection could allow one to use fewer *cNodes* so as to prolong network lifetime.



Conclusion

Proposed solution

cNodes monitor neighbor nodes and apply rules to detect compromised sensors; the set of *cNodes* is periodically renewed:

- *cNodes* selection is based on previous observation by former *cNodes* so as to designate sane (not compromised) nodes with high residual energy
- Coverage issue is addressed by ensuring all nodes are monitored by at least two *cNodes*
- Additional criteria may be used for picking the *cNodes* from set of eligible nodes

Results

- Some overhead and higher global energy consumption
- Better repartition of energy consumption between sensors

Future work

What to do now

- Keep searching for other/better ways to select the *cNodes*
- Run more simulation scenarios (*e.g.* cluster with areas of different activity levels; different network topology…)
- Implement application on a physical testbed (real sensors + TinyOS)
- Use formal modeling (*e.g.* model checking)

Future work

What to do now

- Keep searching for other/better ways to select the *cNodes*
- Run more simulation scenarios (*e.g.* cluster with areas of different activity levels; different network topology…)
- Implement application on a physical testbed (real sensors + TinyOS)
- Use formal modeling (*e.g.* model checking)

Thank you for listening!

Questions