Energy-balancing method to detect denial of service attacks in wireless sensor networks

Quentin Monnet[1] Lynda Mokdad[1] Jalel Ben-Othman[2]

[1]LACL
Université Paris-Est (FR)

[2]L2TI
Université Paris 13 (FR)

COMMUNICATIONS:
CENTREPOINT OF THE DIGITAL ECONOMY
10-14 JUNE 2014 // SYDNEY, AUSTRALIA
#IEEEICC
IEEE
INTERNATIONAL CONFERENCE ON COMMUNICATIONS
ICC 2014
WWW.IEEE-ICC.ORG

speaker:	Pr Lynda Mokdad
contact:	`lynda.mokdad@u-pec.fr`

Outline

Outline

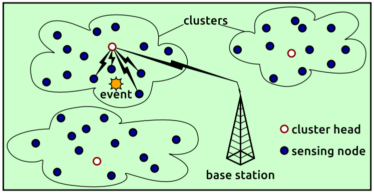
Wireless Sensor Networks (WSNs)

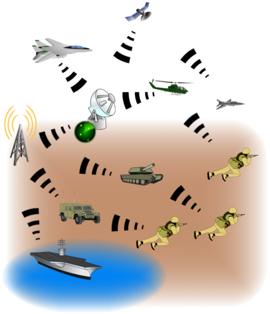
Small devices

- 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

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 lead 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.

Our solution: energy-based *cNodes* selection

New selection process: 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.

Our solution: energy-based *cNodes* selection

New selection process: 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.

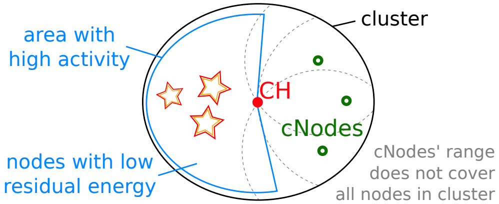
But there are some issues.

Coverage issue

First issue:

The election process is now deterministic. There is no assurance that all sensors can be selected, hence that all nodes will be monitored.

For example: an area with more traffic than in the rest of the cluster.



Complementary *cNodes*

Solution:

When selected, the *cNodes* send the list of their monitored neighbors to the cluster head.

The cluster head should then designate complimentary *cNodes* in case of insufficient coverage.

Security issue

Second issue:

A compromised node can pretend to have a high battery level, even if it is not true, so as to remain *cNode* for all cycles (whereas pseudo-random election process was designed to prevent a given node to remain *cNode* all the time).

A compromised node, if selected as a *cNode*, might escape detection (lower probability to have other *cNodes* around) and can try to fool the cluster head into excluding non-compromised sensors.

Verification nodes

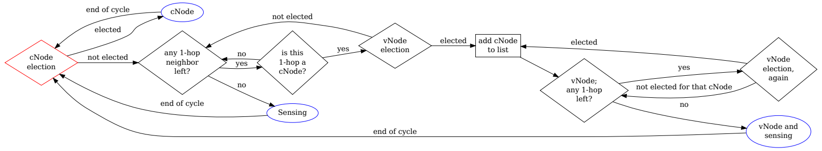
Solution:

Using other nodes (*vNodes*) to verify that *cNodes* do not announce a fake residual energy level.

- *vNodes* are neighbors to newly selected *cNodes*. They self-elect themselves.
- *vNodes* watch *cNodes* and compute an estimation of their residual energy during a cycle.
- They regularly query *cNodes* for their residual energy, and verify that provided values:
  - actually decrease
  - sufficiently decrease, *i.e.* that these values roughly follow the computed consumption estimation
- If *cNodes* keep sending high values over several cycles, they are assumed to try to keep their role and reported to the cluster head

State machine

State machine of the nodes:



Outline

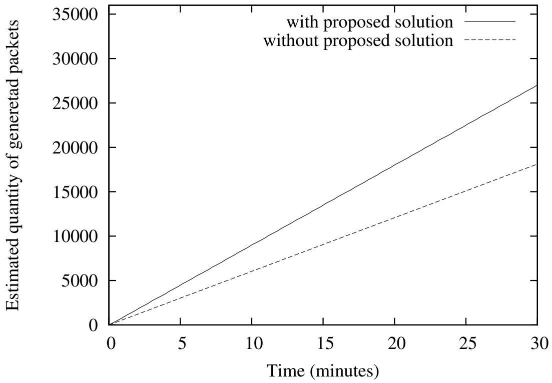
Parameters

We compared energy-driven selection (with *vNodes* use) versus pseudo-random election.
We used ns-3 with following parameters:

Parameter	Value
Number of nodes	30 (plus 1 CH)
Number of *cNodes*	4
Probability for *vNodes* selection	33 %
Delay between consecutive elections	1 minute
Simulation length	30 minutes
Cluster shape	Squared box
Cluster length	Diagonal is 2×50 meters
Transmission range	50 meters
Location of the nodes	CH: center; others: random
Mobility of the nodes	Null
Average data sent by normal nodes	1024 bytes every 3 seconds
Data sent by *vNodes* (per target *cNode*)	1024 bytes every 5 seconds

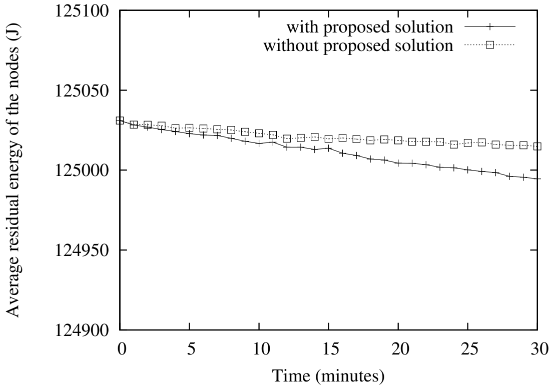
Estimated overhead

Using *vNodes* to regularly query the *cNodes* leads to higher overhead. Below is an estimation of added overhead:



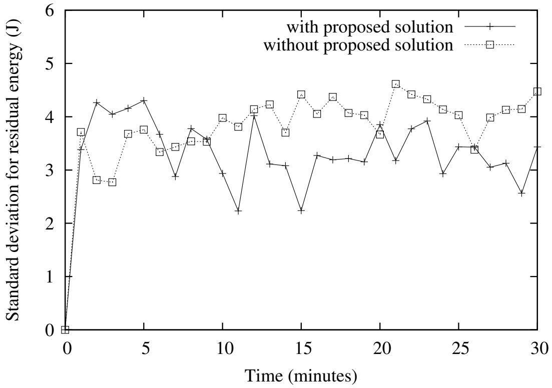
Average consumption

The use of *vNodes* as well as the intervention of cluster head in selection process also leads to a higher average consumption in the network:



Standard deviation for consumption

On the other hand, the load is better balanced between the nodes in the cluster. The standard deviation from average value for consumption is lower for the energy-driven method.



Conclusion

Proposed solution

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

- *cNodes* selection is done according to the amount of residual energy of the sensors
- Coverage issue is addressed by selecting additional *cNodes* when needed
- Security issue is addressed by using *vNodes* which model the energy consumption of *cNodes* and verify that their energy actually decreases “correctly” in regard with computed model

Results

- More overhead and higher global energy consumption
- Better repartition of energy consumption

Future work

What to do now

- More simulation scenarios (*i.e.* cluster with areas of different activity levels)
- Formal modeling (*e.g.* model checking)
- Other ways to select the *cNodes* (*e.g.* democratic election)

The end

Thank you!

Questions?