# Directed Diffusion for Wireless Sensor Networking

**Jussi Nikander**

**Jussi.Nikander@hut.fi**

**9th February 2005**

# Contents

- Directed Diffusion method
- Evaluation
  - Mathematical analysis
  - Simulations
- Conclusion

# Directed Diffusion

- Protocol for *distributed microsensing*, an activity where several cheap, low–powered nodes coordinate to achieve a sensing task
- Designed for robustness, scaling and *energy efficiency*
- Requires only localised interaction between nodes
- Data–centric approach: communication based named data, not named nodes
- four main features: *Interests*, *Gradients*, *Data* and *Reinforcement*

# Interests and Data Naming

- *Interest* is a named task description.
  - Defines the data (sensor events) the originator is interested in
  - Inserted to the network at some (possibly arbitrary) node called a *sink*
  - *Interests* are diffused through the network to all (relevant) nodes
- Naming distinguishes between different tasks
  - Name can, for example, be a number of name–value pairs
- *Exploratory* interests used to find out if there are any interesting phenomena

# Interest Propagation and Storage

- Each node maintains a cache of active *interests*
- an *interest* is a soft state, the *sink* must periodically refresh it
- Each node only knows the previous hop from which it received the interest, not the sink
- Each interest has one or more *gradients* associated with it
- Upon receiving an interest message, a node updates it cache, and may resend the message to (some subset of) its neighbour nodes.

# Gradients

- Each *interest* entry has at least one *gradient* that tells where to forward *data* associated with the *interest*.
- An *interest* may have several *gradients* associated with it
- A *gradient* contains
  - the node where to forward data (the *previous hop* where the *interest* was received from)
  - Data rate, which tells how often data events should be forwarded
  - duration, which tells how long data should be forwarded

# Data Propagation and Path Reinforcement

- Nodes sensing events queried by an *interest* send *data* messages to the network
- Data is forwarded according to the *gradients* associated with the *interest*
- Intermediate nodes may aggregate data in some cases
- When a *sink* starts to received data it can *reinforce* one or more neighbours
  - *Reinforcement* is done to draw in more data
  - Done by sending new interests with larger data rates
  - *Reinforcements* create paths through which data flows at high speed
  - Also possible to negatively reinforce unwanted paths

# Evaluation

- In evaluated situations several source nodes send data messages to several *sinks*.
- Directed Diffusion compared to two idealised schemes
  - *Flooding*, where each data message is sent to every node in the network. Watermark comparison: useful scheme should be at least as good.
  - *Omniscient Multicast*, where each message is sent to the sinks using shortest path multicast trees. Represent best case for IP–networks.

# Mathematical Analysis

- Schemes evaluated in a grid–shaped idealised network with $N$ nodes.
- Results, when $n$ is the number of sources, each of which sent one data message and $m$ number of sinks:
- Total delivery costs are compared
  - Flooding $O(nN)$
  - Omniscient Multicast $O(n\sqrt{N})$, when $m \ll \sqrt{N}$
  - Directed Diffusion $O(n\sqrt{N})$, when $m \ll \sqrt{N}$
- Directed Diffusion more efficient than Omniscient Multicast despite similar results

# Simulation Environment

- Five sizes of networks, between 50 and 250 nodes
- Average node density constant in all simulations
- Five source and five *sink* nodes
- Low idle–time power dissipation
- Metrics used
  - Average dissipated energy
  - Average delay
  - Distinct–event delivery ratio

# Simulated cases

- Comparative evaluation of different schemes
- Impact of temporary node losses on Directed Diffusion
- Impact of Data aggregation and Negative Reinforcement
- Impact of high idle time power consumption

# Simulation results

- Directed Diffusion has lowest average dissipated energy
- Can handle node failures
- Data aggregation and negative reinforcement enhance performance considerably
- Differences in power consumption disappear if idle–time power consumption is high

# Conclusion

- Implementation for several platforms mentioned
- The question of node mobility?
- Questions?