Sensor Networks

TAG: A Tiny AGregation Service for Ad-Hoc Sensor Networks

Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks

Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks

Synopsis Diffusion for Robust Aggregation in Sensor Networks

Sensor Networks

Characteristics of sensor networks

Here are some characteristics of sensor networks:

- Composed of a large number of cheap nodes

Sensor networking research is about efficiently harnessing the *combined* power of these nodes.

Characteristics of sensor networks

Here are some characteristics of sensor networks:

- Composed of a large number of cheap nodes
- Nodes have sensors (bet you didn't see that one coming)

Sensor networking research is about efficiently harnessing the *combined* power of these nodes.

Characteristics of sensor networks

Here are some characteristics of sensor networks:

- Composed of a large number of cheap nodes
- Nodes have sensors (bet you didn't see that one coming)
- Battery powered: energy consumption is a critical issue

Sensor networking research is about efficiently harnessing the *combined* power of these nodes.

Characteristics of sensor networks

Here are some characteristics of sensor networks:

- Composed of a large number of cheap nodes
- Nodes have sensors (bet you didn't see that one coming)
- Battery powered: energy consumption is a critical issue
- Wireless communication: it's *a lot* more expensive than computing

Sensor networking research is about efficiently harnessing the *combined* power of these nodes.

TAG: A Tiny AGregation Service for Ad-Hoc Sensor Networks

Problem

Performing efficient aggregation over the data collected by a sensor network.

- Using a SQL-based declarative approach to specifying data

Problem

Performing efficient aggregation over the data collected by a sensor network.

- Using a SQL-based declarative approach to specifying data
- Treating sensor data as an SQL table

Problem

Performing efficient aggregation over the data collected by a sensor network.

- Using a SQL-based declarative approach to specifying data
- Treating sensor data as an SQL table
- Performing as much in-network computing as possible, to save data transmission

Solution

A query originates from one of the nodes. What now?

- Use query broadcasting to organise network into a tree (rooted at the source)

The idea is that with good time-keeping, intermediate nodes get all answers from their children, aggregate them, and forward the smaller aggregation up-tree.

Solution

A query originates from one of the nodes. What now?

- Use query broadcasting to organise network into a tree (rooted at the source)
- Synchronise all the nodes in order to form time epochs

The idea is that with good time-keeping, intermediate nodes get all answers from their children, aggregate them, and forward the smaller aggregation up-tree.

Solution

A query originates from one of the nodes. What now?

- Use query broadcasting to organise network into a tree (rooted at the source)
- Synchronise all the nodes in order to form time epochs
- At each epoch, leaf nodes propagate sensor readings to their parents

The idea is that with good time-keeping, intermediate nodes get all answers from their children, aggregate them, and forward the smaller aggregation up-tree.

Solution

A query originates from one of the nodes. What now?

- Use query broadcasting to organise network into a tree (rooted at the source)
- Synchronise all the nodes in order to form time epochs
- At each epoch, leaf nodes propagate sensor readings to their parents
- Parents aggregate and do the same

The idea is that with good time-keeping, intermediate nodes get all answers from their children, aggregate them, and forward the smaller aggregation up-tree.

Solution

- Optimisations can be implemented by avoiding unnecessary broadcast for certain aggregations (e.g. min, max), or using multiple parents

Solution

- Optimisations can be implemented by avoiding unnecessary broadcast for certain aggregations (e.g. min, max), or using multiple parents
- Epochs guarantee large amount of wireless downtime for each node

Solution

- Optimisations can be implemented by avoiding unnecessary broadcast for certain aggregations (e.g. min, max), or using multiple parents
- Epochs guarantee large amount of wireless downtime for each node
- Copes very well with network dynamics by simply reforming tree with queries and timeouts

Solution

- Optimisations can be implemented by avoiding unnecessary broadcast for certain aggregations (e.g. min, max), or using multiple parents
- Epochs guarantee large amount of wireless downtime for each node
- Copes very well with network dynamics by simply reforming tree with queries and timeouts
- Copes well with lossy links by maintaining sensor cache.

Solution

- Optimisations can be implemented by avoiding unnecessary broadcast for certain aggregations (e.g. min, max), or using multiple parents
- Epochs guarantee large amount of wireless downtime for each node
- Copes very well with network dynamics by simply reforming tree with queries and timeouts
- Copes well with lossy links by maintaining sensor cache.
- Copes with groups by sending them up-tree if short on memory (increases network traffic)

Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks

Problem

How to efficiently route sensor data to one or more sinks.

- Using variable sensor data-rates
- From only a subset of the sensors with certain localisation properties

Solution

- A request begins by broadcasting an *interest* from the sink across the network

Solution

- A request begins by broadcasting an *interest* from the sink across the network
- Sensors cache which neighbour the request came from

Solution

- A request begins by broadcasting an *interest* from the sink across the network
- Sensors cache which neighbour the request came from
- Sensors matching the location properties of the interest send back data

Solution

- A request begins by broadcasting an *interest* from the sink across the network
- Sensors cache which neighbour the request came from
- Sensors matching the location properties of the interest send back data
- Data gets propagated back towards the sink

Solution

- A request begins by broadcasting an *interest* from the sink across the network
- Sensors cache which neighbour the request came from
- Sensors matching the location properties of the interest send back data
- Data gets propagated back towards the sink
- Using on metrics like which neighbour is sending the most recent updates, the sink reinforces requests from certain neighbours

Solution

- A request begins by broadcasting an *interest* from the sink across the network
- Sensors cache which neighbour the request came from
- Sensors matching the location properties of the interest send back data
- Data gets propagated back towards the sink
- Using on metrics like which neighbour is sending the most recent updates, the sink reinforces requests from certain neighbours
- Neighbours in turn reinforce their own smaller paths, etc

Solution

- A request begins by broadcasting an *interest* from the sink across the network
- Sensors cache which neighbour the request came from
- Sensors matching the location properties of the interest send back data
- Data gets propagated back towards the sink
- Using on metrics like which neighbour is sending the most recent updates, the sink reinforces requests from certain neighbours
- Neighbours in turn reinforce their own smaller paths, etc
- Explicit negative reinforcement is applied to previous connections behaving badly

Solution

- Selects few high-performance paths for high data-rate, the rest remain idle

Solution

- Selects few high-performance paths for high data-rate, the rest remain idle
- Adapts to dynamic network conditions by reinforcing good neighbours and negatively reinforcing bad neighbours

Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks

Problem

Update sensor node code at dynamically in a sensor network.

- A good compromise between update speed and energy consumption
- Unsynchronised nodes

Solution

- Every node knows metadata about its own code, indicating its version, and interval size

Solution

- Every node knows metadata about its own code, indicating its version, and interval size
- Every interval, a node broadcasts its code and doubles interval size (up to a max)

Solution

- Every node knows metadata about its own code, indicating its version, and interval size
- Every interval, a node broadcasts its code and doubles interval size (up to a max)
- If a node hears a broadcast with newer code, it adopts the code, rebroadcasts it and reduces interval size (to min)

Solution

- Every node knows metadata about its own code, indicating its version, and interval size
- Every interval, a node broadcasts its code and doubles interval size (up to a max)
- If a node hears a broadcast with newer code, it adopts the code, rebroadcasts it and reduces interval size (to min)
- If a node hears a broadcast with older code, it broadcasts its own code

Solution

- Every node knows metadata about its own code, indicating its version, and interval size
- Every interval, a node broadcasts its code and doubles interval size (up to a max)
- If a node hears a broadcast with newer code, it adopts the code, rebroadcasts it and reduces interval size (to min)
- If a node hears a broadcast with older code, it broadcasts its own code
- If a node hears a broadcast with its own code, it increments a counter.

Solution

- Every node knows metadata about its own code, indicating its version, and interval size
- Every interval, a node broadcasts its code and doubles interval size (up to a max)
- If a node hears a broadcast with newer code, it adopts the code, rebroadcasts it and reduces interval size (to min)
- If a node hears a broadcast with older code, it broadcasts its own code
- If a node hears a broadcast with its own code, it increments a counter.
- If within an interval the counter exceeds a given threshold, the node scraps its transmission

Solution

- Achieves adaptive update speed by changing intervals

Solution

- Achieves adaptive update speed by changing intervals
- When new code is detected, all nodes have small intervals, retransmitting fast to ensure new code reaches everyone

Solution

- Achieves adaptive update speed by changing intervals
- When new code is detected, all nodes have small intervals, retransmitting fast to ensure new code reaches everyone
- Once new code has been propagated, transmissions gets cut because both intervals and counters increase

Solution

- Achieves adaptive update speed by changing intervals
- When new code is detected, all nodes have small intervals, retransmitting fast to ensure new code reaches everyone
- Once new code has been propagated, transmissions gets cut because both intervals and counters increase
- In a stable network, only a subset of nodes will transmit, while the others remain silent

Synopsis Diffusion for Robust Aggregation in Sensor Networks

Problem

Same as TAG!

Solution

- The main problem with TAG was using a tree topology

Solution

- The main problem with TAG was using a tree topology
- This was virtually enforced by having duplicate-sensitive aggregates

Solution

- The main problem with TAG was using a tree topology
- This was virtually enforced by having duplicate-sensitive aggregates
- It turns out that it's possible to encode aggregates as order- and duplicate-insensitive synopses (e.g. coin toss!)

Solution

- The main problem with TAG was using a tree topology
- This was virtually enforced by having duplicate-sensitive aggregates
- It turns out that it's possible to encode aggregates as order- and duplicate-insensitive synopses (e.g. coin toss!)
- Synopses can be routed through a ring-based graph to the query source

Solution

- Because of a graph topology, the proportion of nodes participating in a query is *much* higher

Solution

- Because of a graph topology, the proportion of nodes participating in a query is *much* higher
- There are strong guarantees on the error estimates for the result of these synopses:

Solution

- Because of a graph topology, the proportion of nodes participating in a query is *much* higher
- There are strong guarantees on the error estimates for the result of these synopses:
  - All nodes that communicated successfully through at least one path are included

Solution

- Because of a graph topology, the proportion of nodes participating in a query is *much* higher
- There are strong guarantees on the error estimates for the result of these synopses:
    - All nodes that communicated successfully through at least one path are included
    - The result is the same as that of applying the synopse functions to a single datastream containing all the data