To be defined

Authors

***Abstract*—**
Abstract to be written.

I. Introduction

In what follows we shall make a short presentation regarding the following aspects related with wireless sensor networks, and applicable in the context of our project:

- we shall tacle the issues regarding the architectural models and interaction paradigms used in sensor networks;
- we describe (one of the few) protocol that addresses the practical issues of communication, meaning the packing and flow of messages exchanged between the different actors; (this is at the application layer in the OSI model;)
- (step downwards to the transport / network) as we make an overview related with existing network stacks in two of the most known operating systems (targeting sensor networks): Contiky and TinyOS;

A. Sensor query protocols and models

During the initial phase of the project we tried to determine which are the most appropriate (and the most used) models and interaction paradigms inside a (wireless) sensor network.

1) Models: Thus we started with the models involved, which — generally speaking — are referring to:

- which are the entities, or as we call them the actors involved (or better said the classes of actors); (we shall see in a minute about them;)
- the context or environment in which the actors execute their designated tasks; (for example we could be speaking here about the limited power source, the limited or high error-rate communication medium, or even about the remote locations in which they operate;)
- the operational constraints that result from the environment; (for example the limited power source implies that the communications should be kept at a minimum, buffering as much data until a full package could be sent; or the fact that the actors should be autonomous and self-maintained because human-based maintainance on the site is not possible due to the remote location;)
- the way in which the responsabilities (the tasks) are divided and assigned to different actors; (some of them might have special roles;)
- activity classes — generally speaking which are the most encountered types of tasks;

The result of the survey we concluded that almost all read papers split the actors in mainly three classes:

- the frontend — a central entity, usually a powerfull (or at least resonably powerfull) device, that allows final users to interact with the sensor network; they usually present a user interface, or allow the collected data to be exported in a certain format;
- gateway — it resembles the frontend, with the exception that this one is not user-centric, but it serves as a mediator between the sensor network and other applications; it usually exports an network API (through a binary protocol);
- sensor or actuator — the actual sensing device;

As a consequence we found this generaly aggreed-upon model is well suited to our needs, and we didn't investigate further on this matter.

Regarding the activity classes one paper [**?**] gives a high lewel classification:

- querying — it is described as a synchronous the process in which a central unit (a front-end or a gateway) generates an exact and explicit data query and 'injects' it into the network, expecting that each node will execute it and send back the results in order for the central unit to interpret the data;
- tasking — a coordinated activity between nodes that could span longer periods of time (more than a couple of minutes); the sensors are considered to be active and autonomous;

For the moment, in the context of the Dehems project, we are interested mainly on querying operations (sensor readings), but in the future we could also turn them into active (decision and action capable) devices (actuators), and thus focus on tasking operations.

Thus going further on the querying operations path, another paper [**?**] classifies them as:

- pull (active) querying — the central unit asks (by means of area multicast or broadcast) all the sensors in (a part of the) network for certain readings; and as the readings arrive back, they are aggregated;
- push (passive) querying — the central unit asks (again either the entire or a part of the network) to be notified when a certain reading is modified; then it waits and aggregates continuously the received data;
- combined;

They also note that active querying is usable in short period, direct interest, not frequent queries; meanwhile the passive querying should be used for long period, continuously monitorying of readings.

As querying paradigms, almost all read papers (concerning this problem) [**?**], [**?**], [**?**], [**?**] propose a SQL like interface, that allows users to write specific queries that will be executed against the sensor network. Also almost all of them allow both push and pull type of queries to be written.

Unfortunatly these solutions (and their backing execution platforms) have at least some disadvantage, that in the context of the Dehems project renders them unusable:

- they can't be used in data-mining tasks, because they were not designed to collect data, but instead to efficiently aggregate data close to the node, and propagate through the network only partial results; and in data-mining tasks we need exactly the oposite: we want to collect as much raw data as possible, to be able to process it afterwards.
- the query must have a precise goal and can't be addapted to a dynamic sensor network environment;

In our research we haven't found a proposal for a framework that would be focused on data aquisition. Althought we did found solutions for querying data streams (like [**?**]).

2) Interaction: Regarding the interaction paradigms we are referring to the following issues:

- actor identification — how do actors know to disdinguish between multiple instances of the same class; (also how do we establish a long term identification scheme;)
- actor addressing (or better said node addressing) — the way in which we actually tagg the identity of an actor with an actual message delivery address; (altought the addressing resembles the identification, the current one has a more transient nature, beeing short termed, and usually dependent on external factors; think for example at the distinction between a laptop (its identity given by the owner) and the IP address assigned differently as the laptop roams from a network to another;)
- actor discovery — the method used by any actor to discover nearby actors, or actors that could offer needed services;
- actor interaction — how different actors exchange information one with another;

Regarding the first two issues (identification and addressing) there is not to much in the litreture, usually identification beeing assimilated with addressing, and addressing beeing solved by a number assigned at production time (like the MAC in the case of Ethernet).

Now because in the context of our project we must be able to identify on the long term each sensor, but because sensors could travel from one location (household) into another, we should be able to decouple the identification (which should pe permanent) and the addressing (which could be based on the current location).

Also usually discovery is reduced to finding the addresses of all the sensors (actors) inside a network and their capabilities. (More about this in the next paragraph.)

About the actual interaction model, one paper [**?**] splits them into:

- unicast — from one actor to exactly another (exactly) identified actor;
- area multicast — from one actor to all the actors inside an (identified) physical area;
- area anycast — from one actor to (exactly) one actor inside a physical area;
- broadcast — from one actor to all the other actors inside a network;

Going even further [**?**] proposes a different view:

- abc — 'anonymous best-effort single-hop broadcast' — sends the same message to all the direct neighbours (unreliable);
- ibc — 'identified best-effort single-hop broadcast' — sends the same message (but attached with a senders's address) to all the direct neighbours (unreliable);
- uc — 'best-effort single-hop unicast' — sends one message to a designated direct neighbour (sender and receiver attached addresses, unreliable);
- suc — 'stubborn single-hop unicast' — sends the same message repeatedly to the same designated direct neighbour (sender and receiver attached addresses, unreliable, continuous);
- ruc — 'reliable single-hop unicast' — sends one message to a designated direct neighbour (sender and receiver attached addresses, reliable);
- polite — 'polite single-hop broadcast' — sends

B. SSI – Simple Sensor Interface

SSI is a network protocol described in [**?**] that allows a device to query sensor devices, it is developed within the european project MIMOSA, and in what follows we shall make a short presentation regarding it based on the previously cited paper.

The SSI protocol is intended to be used for short range wireless sensor networks, or for sensors directly connected to the querying device (the terminal as they call it, or gateway / frontend as it is called in our terminology), and its design principles were:

- simplicity — because the devices must run for long periods of time with small batteries or limited power sources, the protocol must not require to much processing power (which leads to power consumption);
- communication links could be shared — more than one node could be connected on the same communication medium, as it allways happens in a wireless environment, or as it could happen if the communication is done over the electrical power grid;
- multiple sensors could be hosted inside a single node;
- node discovery — the protocol should be addapted to a dynamic sensor network, and should provide means for discovery of both the participating nodes and their abilities;
- the protocol should allow both data aquisition, and node configuration tasks;

We have listed these principles here, because they apply almost entirely to the context of our own project. (Except maybe the fact that we could indulge ourselves to a little more complex solution, because our target are medium sensor nodes (see targeted devices section ???), and their solution targets low-level sensor boards.)

The protocol implies the exchange of small packets, and we shall only review the possible operations:

- query and discover commands — that allows the terminal (gateway) to discover both nodes and their attached sensors; also it allows some parameters to be discovered (as implemented protocol version, sensor types, identifier (address), unit or scale, etc.);

- get / set configuration and reset — used to control the nodes operating parameters;
- request data — used to implement active (pull) querying;
- create / destroy observer — allowing the terminal to be notified about sensor data changes, thus allowing the passive (push) querying paradigm; what is also important to note is that the observers could be created at the request of the terminal, but also at the request of the sensor node;