Demonstration of the Sensor Data Transmission and Management Protocol (STMP)

Nils Aschenbruck°•, Christoph Fuchs°•, Sascha Jopen°, Tim Schneider°
° University of Bonn - Institute of Computer Science 4, Friedrich-Ebert-Allee 144, 53113 Bonn, Germany
• Fraunhofer FKIE, Neuenahrer Str. 20, 53343 Wachtberg, Germany
{aschenbruck, cf, jopen, schneid5}@cs.uni-bonn.de

I. Motivation

Nowadays, networked sensors are deployed in various scenarios. To transmit sensor data like GPS, temperature, or vital signs data robust communication networks are needed. Different applications and sensor types may have specific requirements to the transmission of their data regarding loss-tolerance, timeliness, priority, or completeness. These requirements are typically met by appropriate transport protocols. At the lower layers, heterogeneous networks like wireless mesh networks or long thin networks lead to specific channel characteristics that may require the use of specifically adapted transport protocols. Additionally, the type of scenario can also affect the transport layer, e.g., a scenario with static nodes compared to moving nodes with high mobility.

Of course, there are many variable requirements of different sensor data streams and applications. But there are also some basic functionalities that are common, such as registering sensors at a receiver, timestamping of sensor data as well as synchronization of all nodes.

Thus, we specify a new protocol for sensor data transmission and management (STMP) that provides the basic functionalities required by many sensor data streams while at the same time allowing a flexible choice of the transport protocol to be used.

The rest of this abstract is organized as follows: In section II we introduce the basic idea of STMP. Next, we describe our implementation in the tool called *BonnSens* (Section III). Finally, we explain the setup of the demonstration IV.

II. Sensor Data Transport and Management Protocol (STMP)

In order to avoid the necessity to implement these common functionalities anew for each sensor data application and with the intention to provide a common framework for the transport of sensor data we specified the *Sensor Data Transport and Management Protocol (STMP)* [3]. Besides the transport of sensor data from a sensor node to the fusion point, STMP is also designed for providing sensor management functions to the fusion point. These functions include for example adaptation of a sensor's sample rate or measurement precision, sensor polling, or other runtime re-configurations of the sensor nodes. For further details, see [3].

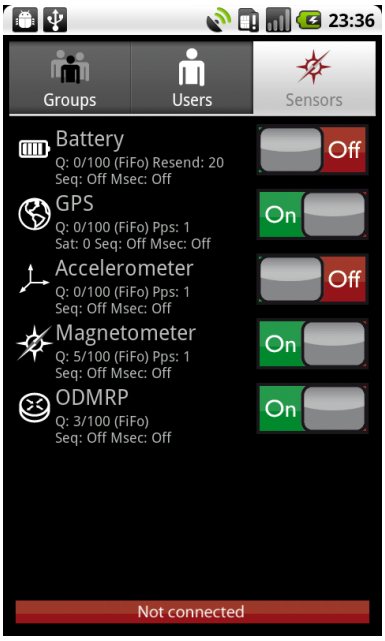

Fig. 1. Screenshot of the BonnSens client application for Android.

III. BonnSens

In order to demonstrate the feasibility of STMP and to take measurements in real scenarios, we implemented STMP and a sensor data client application for the Android platform. Android [1] is an open source operating system developed by Google for smartphone devices. As modern smartphones are already equipped with a number of sensors like GPS, accelerometers, magnetometers and also come with an IEEE 802.11 [2] wireless network interface, they provide an ideal hardware platform for a wireless sensor node. The Android operating system is Unix-based and, thus, allows for a limited portability of certain software modules to other Unix derivates running on a PC or laptop. In order to realize a consistent implementation of STMP for sensor data applications running on smartphones as well as for applications hosted on a laptop, we modularized STMP by implementing it as a commonly usable library in C named *libSTMP*. Client applications may thus be programmed either using the Android API for the deployment on smartphones or using native C code for the deployment on a laptop both accessing the same STMP

library. In this initial implementation, we used UDP at the transport layer. We will integrate further transport protocols in the future. Figure 1 shows a screenshot of our sensor data client application *BonnSens* for Android. BonnSens currently supports the collection and transmission of battery, GPS, accelerometer, and magnetometer data as well as neighbor metric data from the ODMRP [4] routing protocol. Each sensor at a client can be enabled separately. On activation, a client creates an STMP sensor data association with the fusion point by registering its active sensors. Several configurable options are available for each sensor like the interval for sending data and the usage of a LIFO queuing scheme instead of the standard FIFO queue.

On the server side, that is the fusion point, we implemented a server application using the STMP library for receiving the sensor data from the clients and for writing it to a database. The sensor data fusion algorithm directly operates on the data stored in the database. The clients' tracks can be displayed using a tracking GUI.

IV. Demo Setup

We will demonstrate the implementation of STMP within BonnSens as well as the tracking GUI. This demonstration complements our full paper [3] in the main track of LCN 2011.

Acknowledgments

This work was supported in part by CONET, the Cooperating Objects Network of Excellence, funded by the European Commission under FP7 with contract number FP7-2007-2-224053.

References

[1] "Android Developer Homepage," 2011. [Online]. Available: http://developer.android.com/index.html

[2] "Homepage of IEEE 802.11 Working Group," 2011. [Online]. Available: http://grouper.ieee.org/groups/802/11

[3] N. Aschenbruck and C. Fuchs, "STMP - Sensor Data Transmission and Management Protocol," in *Proc. of the 36th IEEE Conference on Local Computer Networks*, 2011.

[4] S.-J. Lee, M. Gerla, and C.-C. Chiang, "On-demand multicast routing protocol," in *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*, 1999, pp. 1298–1302.