Poster: Model-Based Real Time Analysis of Distributed Human Activity Recognition Stages in Wireless Sensor Networks

Florian Grützmacher
University of Rostock
Rostock, Germany
florian.gruetzmacher@uni-rostock.de

Benjamin Beichler
University of Rostock
Rostock, Germany

Christian Haubelt
University of Rostock
Rostock, Germany

ABSTRACT

In the paper at hand, a model-based design and energy estimation approach for wireless sensor nodes in human activity recognition systems is extended. Entire wireless body area sensor networks are modeled and analyzed w.r.t. their real time capabilities of different software mappings on a system level.

CCS CONCEPTS

• **Networks** → **Network performance analysis**; • **Computer systems organization** → *Real-time system architecture*.

KEYWORDS

Activity Recognition; Wireless Sensors; Real-Time Analysis

ACM Reference Format:
Florian Grützmacher, Benjamin Beichler, and Christian Haubelt. 2019. Poster: Model-Based Real Time Analysis of Distributed Human Activity Recognition Stages in Wireless Sensor Networks. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2019 International Symposium on Wearable Computers (UbiComp/ISWC '19 Adjunct), September 9–13, 2019, London, United Kingdom.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3341162.3343845

1 INTRODUCTION AND RELATED WORK

Current generations of sensors on the market already integrate computing capabilities to perform sensor data processing on-chip, allowing for sophisticated edge computing systems [2]. This enables the development of smart sensors for ubiquitous mobile sensing systems, resulting in heterogeneous multi-processing architectures spanning over, e.g., *Wireless Sensor Networks* (WSN). The complexity of such architectures results in an increased number of design decisions that have to be made during development of mobile ubiquitous systems. Design decisions include mapping the software onto available processing units of the WSN, influencing different design goals like real time performance, latency or energy consumption.

The most similar work to the paper at hand is [4], introducing a model-based design and energy estimation approach of wireless sensors for activity recognition scenarios. While in [4] the modeling approach is shown for a single wireless sensor node, the paper at hand extends it by modeling an entire WSN. Furthermore, an extension for a model-based real time analysis of the WSN is introduced.

UbiComp/ISWC '19 Adjunct, September 9–13, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6869-8/19/09.
https://doi.org/10.1145/3341162.3343845

2 CASE STUDY

As a case study, we used the kitchen assessment scenario introduced in [5]. Here, accelerometer and gyroscope from five sensors attached to a human body where sampled at a rate of 120 Hz. A fixed size sliding window of 128 samples is used to segment the data with an overlap of 75 %. From each window, four statistical features were calculated, i.e., *mean*, *variance*, *skewness*, and *kurtosis*, as well as two frequency domain features computed by a *Fast Fourier Transform* (*FFT*), i.e., the dominant frequency and its magnitude (in [5] referred to as *peak* and *energy*, respectively).

The aforementioned offline scenario was selected as a case study for our work, designing it towards an online system. We chose custom energy-efficient sensor nodes to deploy a WSN. They are equipped with a BHI160 ultra-low-power sensor hub [2] composed of an accelerometer, a gyroscope, and a 32-bit floating point optimized microcontroller referred to as *FuserCore*. Furthermore, the sensor nodes are equipped with a DA14583 microcontroller with an integrated Bluetooth Low Energy (BTLE) stack and radio for data transmission. As the BHI160 does not allow a sampling rate of 120 Hz, we chose the nearest possible sampling frequency, i.e., 100 Hz. In order to analyze the real time capabilities of the WSN in our setup, we use a model-based representation on which the real time analysis is done.

3 BACKGROUND

The basis of the modeling approach are *Cyclo-Static Data Flow* (*CSDF*) graphs [1], an extension of *Synchronous Dataflow* (*SDF*) graphs as described by [6].

Definition 1. A *CSDF graph* $G = (V, E, cons, prod, D)$ consists of a set of vertices V called actors, a set of edges $E \subseteq V \times V$ called channels, token consumption rate vectors $cons : E \longrightarrow \mathbb{N}^n$, token production rate vectors $prod : E \longrightarrow \mathbb{N}^n$, and an initial token distribution $D : E \longrightarrow \mathbb{N}$.

In CSDF, each actor fires in cyclic repetitive sequences of length n, called *cycles*. The function $n : V \longrightarrow \mathbb{N}$ assigns a number of firings per cycle to each actor. The presence and number of data packets is represented by tokens. With each channel e, a number of *initial tokens* $D(e)$ is associated. The execution of a consistent CSDF graph forms fixed repetitive firing sequences called *iterations*, which can be described by a *repetition vector* γ.

Definition 2. The *repetition vector* γ of a CSDF graph G assigns a number of firings per iteration to each actor γ: $V \longrightarrow \mathbb{N}$. It is defined as the unique smallest non-trivial vector that satisfies the following balance equation:

$$\sum_{i=0}^{\gamma(\widetilde{v})-1} prod(e)[i \bmod n(\widetilde{v})] = \sum_{j=0}^{\gamma(v)-1} cons(e)[j \bmod n(v)]$$

for each channel $e = (\widetilde{v}, v) \in E$ from actor $\widetilde{v} \in V$ to $v \in V$. In this sense, non-trivial means that $\gamma(v) > 0$ for all $v \in V$.

Furthermore, we are using timed CSDF graphs [8], where an n-dimensional execution time vector δ, also referred to as *delay* is associated with each actor, i.e., $\delta : V \longrightarrow \mathbb{N}^n$. This allows to calculate the throughput as a further important property of a CSDF graph. According to [8] it is defined as follows.

Definition 3. The *throughput* $TH(G)$ of a CSDF graph G is defined as the average number of iterations of a time period divided by the duration of that period, i.e., the reciprocal of the average time duration of an iteration of the graph G.

4 DATAFLOW-BASED REAL TIME ANALYSIS

In this section, we shortly present the modeling of WSNs in Human Activity Recognition (HAR) scenarios. The foundation for a single sensor can be found in [4].

We modeled the selected case study according to the Activity Recognition Chain from [3]. The data acquisition is modeled by actors `DAx` and pre-processing (built-in sensor firmware) by actors `PPx`. The segmentation and feature extraction are summarized in actors `FEx`, calculating the window-based features. We summarized the Modeling & Inference stage and the classification into a placeholder actor `CL` as it is not part of the analysis. The corresponding

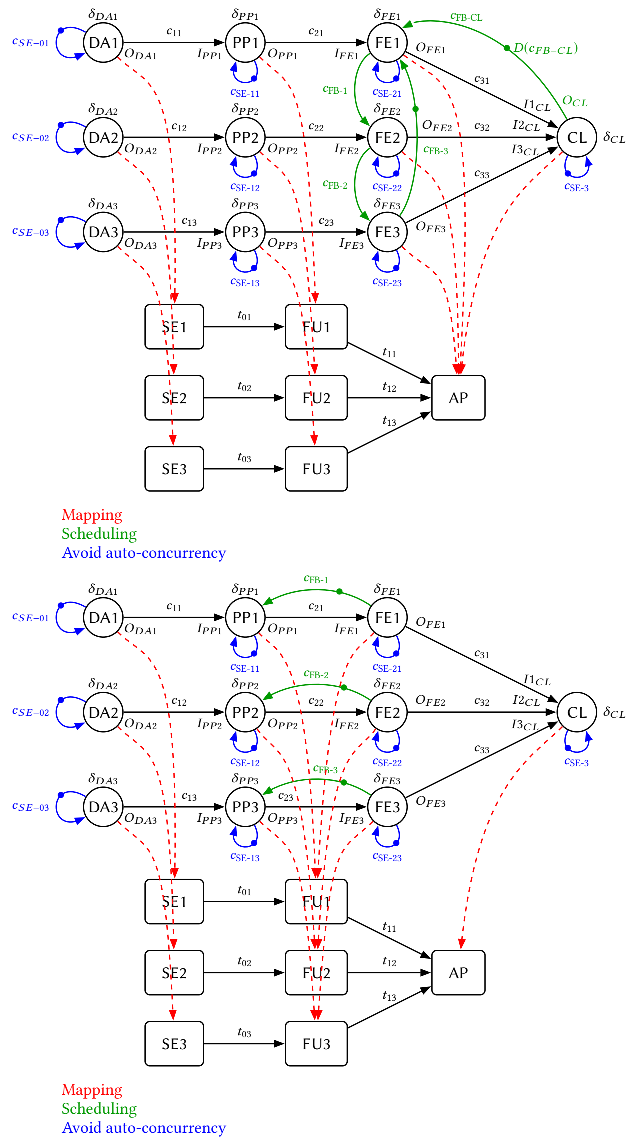

Figure 1: CSDF graph in Mapping A (top) and B (bottom)

CSDF graph is shown in Figure 1. Due to space limitations, only 3 out of 5 sensors are shown. Initial tokens are marked by bullets. The corresponding parameters of the CSDF graph are calculated from the configuration of the activity recognition system, i.e., sampling rate, window size, feature vectors, etc., and are shown in Table 1. Note that '. . . ' indicates recurrence of previous vector entries throughout the paper.

A *hardware model* $H = (P, T, S)$ is a set of processing cores P, a set of sensors S, and a set of directly connected communication channels $T \subseteq S \times P \cup P \times P$.

In our experiments, each wireless sensor node consists of a sensor named SE, the FuserCore named FU, a wired

Table 1: Graph parameters of all configurations

Graph Parameters	Parameter Values
δ_{DAx}	$10\,000\mu s$
δ_{PPx}	$1\,250\mu s$
$\delta_{FEx} \in \mathbb{N}^{32}$	$[376\mu s, 344\mu s, \cdots, 23\,469\mu s]$
$\delta_{CL} \in \mathbb{N}^{32}$	$[0\mu s, \cdots, 1\,000\mu s]$
$O_{DAx}, I_{PPx}, O_{PPx}, I_{FEx}$	2
$O_{FEx}, Ix_{CL} \in \mathbb{N}^{32}$	$[0, \cdots, 12]$
$O_{CL} \in \mathbb{N}^{32}$	$[0, \cdots, 32]$
$D(c_{FB-CL})$	32

communication link, i.e., I^2C between them (t_0), the smartphone's application processor named AP, and the BTLE link t_1 between them.

A *mapping* M consists of a set of mapping relations $M \subseteq V \times (S \cup P)$, restricting each actor v to be mapped exactly once, i.e., $\forall v \in V : !\exists m_V = (v, p) \in M$. After mapping, scheduling edges need to be added to the graph, in order to serialize the executions of all actors mapped to the same processor in a non-overlapping way.

Different mappings of the application graph to the hardware are possible. In Figure 1, two different mappings of the CSDF graph including the corresponding scheduling edges are shown. In `Mapping A`, the raw data from all sensors is sent to the smartphone, where segmentation, feature extraction, and classification are performed. In `Mapping B`, segmentation and feature extraction are performed on each sensor's *FuserCore*. Only the feature vectors are sent to the smartphone where classification is performed.

In order to decide which mapping is beneficial in terms of real time performance for the BTLE transceiver of the smartphone (i.e., whether the data rate meets the BTLE transmission constraints of the smartphone), the effort function $ef_{trans}(\tilde{t})$ capturing the transmission amount on a set of hardware links $\tilde{t}$ calculates its resulting data rate. The data rate can be calculated by summing up all consumption rates on the edges mapped to the links $\tilde{t}$ within one graph iteration, and dividing it by the average duration of one graph iteration or multiplying it by its throughput, respectively. In Algorithm 1 the pseudo code for this calculation is shown.

A set of hardware channels $\tilde{t}$ is given to the algorithm, which, in our case, are the links between the sensors and AP (t_{11}-t_{15}) representing the BTLE connections. Furthermore the graph G, the hardware model H, the mapping M, the repetition vector $\gamma(G)$, and the graphs throughput $TH(G)$ are provided as parameter. For each hardware channel (line 3), the graph edges $\tilde{E}$ mapped to it are returned by function $getMappedChannels()$. For each edge, the destination actor's repetition vector entry is saved in a variable r (lines 6 and 7). Iterating over all firings of that actor, the consumption rates of its incoming channel are summed up (line 8 and 9). Repeating that for all hardware channels $\tilde{t}$ results in the accumulated token consumptions (variable *sum*) of all actors mapped to AP within one graph iteration, representing the received BTLE packets on AP. Finally, in line 10 the data rate is calculated and returned.

Algorithm 1 Calculating effort function $ef_{trans}(\tilde{t})$.

1: **procedure** CALC_EFFORT($\tilde{t}$,G,H, M,$\gamma(G)$, $TH(G)$)
2: $\quad sum = 0$
3: $\quad$ **for** all t in $\tilde{t}$ **do**
4: $\quad\quad \tilde{E} = getMappedChannels(t, G, H, M)$
5: $\quad\quad$ **for** all $\tilde{e}_i$ in $\tilde{E}$ **do**
6: $\quad\quad\quad \tilde{v}_i = getDestActor(\tilde{e}_i, G)$
7: $\quad\quad\quad r = \gamma(\tilde{v}_i)$
8: $\quad\quad\quad$ **for** all j in $[0, 1, ..., r-1]$ **do**
9: $\quad\quad\quad\quad sum \mathrel{+}= cons(\tilde{e}_i)[j \bmod n(\tilde{v}_i)]$
10: $\quad$ return $sum * TH(G)$

5 EXPERIMENTAL EVALUATION

In our experimental setup, we used a Samsung Galaxy S5 smartphone as the data aggregating device of the WSN. The 5 sensors connected via BTLE were placed in a distance of ≈ 85 cm from the smartphone. In order to empirically determine the maximum data rate of the smartphones BTLE chip, we tried different number of sensors, sampling frequencies and BTLE connection intervals. In all experiments, each sensor sample (3D raw accelerometer, 3D raw gyroscope, or 3D feature vector entry) is packed in a 14 byte data structure. The maximum data rate of ≈ 398.2 samples/s without sample loss could be achieved with 4 sensors, each sampling at 100 Hz and a BTLE connection interval of 30. The results were acquired by counting all received samples from each sensor within a specified time of at least 90 s. Regarding this empirically determined maximum data rate, we analyzed our model in both mappings with five wireless sensor nodes. In `Mapping A`, the wireless sensor nodes send raw accelerometer and gyroscope data each sampled at 100Hz to the smartphone. In `Mapping B`, the window-based feature extraction is performed on the BHI160's FuserCore. The calculated feature vectors from each sensor are transmitted via BTLE to the smartphone. For the delay annotations of each actor, we measured the worst-case execution times performed on the FuserCore. Note, when mapping actors to the smartphone AP, we used the same execution times as on the FuserCore. This can be seen as an overestimation since we expect the smartphone AP to execute the feature extraction code faster than the FuserCore. However, we consider this overestimation as adequate, since the chosen execution delay on the

Table 2: Estimated and measured wireless transmissions of MappingB in *samples/s*

Wireless Link in H	Estimated	Measured	Error
t_{11}	37.5	37.63	0.35%
t_{12}	37.5	37.72	0.59%
t_{13}	37.5	37.26	0.64%
t_{14}	37.5	37.51	0.03%
t_{15}	37.5	37.14	0.96%
$t_{11} - t_{15}$	187.5	187.27	0.12%

AP does not affect the real time performance of the WSN in our model. For the same reason we used a placeholder execution time of 1 000 μs for the classification, modeled by actor CL. We used the SDF^3 tools [7] to analyze the throughput and repetition vector of the resulting graphs from both mappings in Figure 1, but including all five sensors. For both mappings, the same repetition vector $\gamma(G) \in \mathbb{N}^{16} = [32, \ldots]$ and the throughput $TH(G) = 3.125e-06\frac{1}{s}$ was acquired. The transmission efforts of the smartphone's BTLE transceiver calculated by Algorithm 1 for both mappings are 1 000 Tokens/s and 187.5 Tokens/s for `Mapping A` and `Mapping B`, respectively. Note that a token in our model corresponds to one 14 Byte sensor sample/feature value. We can see that `Mapping A` does not meet the real time requirements of the configuration due to the limited date rate of the smartphone's BTLE transceiver ($\approx$ 398.2 samples/s) and is thus not possible to deploy. In `Mapping B`, the data rate on $t_{11} - t_{15}$ is significantly reduced to 187.5 samples/s and meets the data rate constraint of the smartphone's BTLE transceiver. Additionally, the real-time capability of the processing cores can be validated by checking if the sensors constrain the graph's iteration period: $\frac{1}{TH(G)} \stackrel{?}{=} \sum_{i=0}^{\gamma(DAx)-1} \delta_{DAx}[i \bmod n(DAx)]$.

In order to evaluate the model accuracy, we measured the resulting data rates on the smartphones BTLE transceiver after implementing `Mapping B` on the sensors. For this setup, a BTLE connection interval of 30 has been used. The acquired results along with the model-based estimations and errors are shown in Table 2. We can see, that the estimated results from our model differ a maximum of 0.96 % from the measured results in our experiments. The sensors internal clocks typically vary by ±1% [2]. As a result, the measurements from our implementation slightly differ from the model-based estimations. However, the deviations are small enough to allow substantiated design decisions at early design stages.

The results acquired from our model might appear straight forward for our selected case study. However, for the sake of comprehensibility only a subset of mappings and a simple case of an HAR setup was analyzed in our experiments. The introduced approach shows its real potential in scenarios, where the feature extraction is not symmetrical on all sensors, e.g., additional features are calculated on the two sensors at the feet of the person for additional step detection. Another possible scenario is a feature selection approach, to find out the most important features of each body position. This can lead to very different feature sets for each sensor/body part. In such cases, different mappings for sensors are more reasonable and the analysis and conclusion which mapping is the most beneficial becomes difficult for a developer to estimate manually. Furthermore, modeling an entire sensor network is necessary, as a mapping decision for one sensor affects the resulting data rate at the receiver. This in turn influences mapping options of the remaining sensors.

6 CONCLUSION

In the paper at hand, a model-based real time analysis method for different mappings of activity recognition stages onto a wireless body area sensor network is presented and evaluated for a selected case study. In our experiments, our model-based approach achieved a system-level accuracy above 99 %. The generality of the method allows to analyze alternative mappings and can be applied to more complex system designs. Considering the growing system complexities, formal estimation methods are crucial for automated design space exploration, in order to deal with the increasing design space of today's and future human activity recognition systems.

ACKNOWLEDGMENTS

We thank Bosch Sensortec GmbH for providing the required sensor hardware and programming support (BMBF grant number 03ZZ0519G).

REFERENCES

[1] Greet Bilsen, Marc Engels, Rudy Lauwereins, and Jean Peperstraete. 1996. Cycle-static dataflow. *IEEE Transactions on signal processing* 44, 2 (1996), 397–408.

[2] Bosch Sensortec. 2017. BHI160 / BHI160B - Ultra low-power sensor hub incl. integrated imu. BST-BHI160(B)-DS000-01.

[3] Andreas Bulling, Ulf Blanke, and Bernt Schiele. 2014. A Tutorial on Human Activity Recognition Using Body-worn Inertial Sensors. *ACM Computing Surveys (CSUR)* 46, 3, Article 33 (jan 2014), 33 pages.

[4] Florian Grützmacher, Albert Hein, Thomas Kirste, and Christian Haubelt. 2018. Model-Based Design of Energy-Efficient Human Activity Recognition Systems with Wearable Sensors. *Technologies* 6, 4 (2018), 89.

[5] Frank Krüger, Martin Nyolt, Kristina Yordanova, Albert Hein, and Thomas Kirste. 2014. Computational state space models for activity and intention recognition. A feasibility study. *PloS one* 9, 11 (2014), e109381.

[6] E.A. Lee and D.G. Messerschmitt. 1987. Synchronous data flow. *Proc. IEEE* 75, 9 (Sept 1987), 1235–1245.

[7] Sander Stuijk, Marc Geilen, and Twan Basten. 2006. SDFˆ 3: SDF for free. In *Proceedings of ACSD'06*. IEEE, 276–278.

[8] Sander Stuijk, Marc Geilen, and Twan Basten. 2008. Throughput-buffering trade-off exploration for cyclo-static and synchronous dataflow graphs. *IEEE Trans. Comput.* 57, 10 (2008), 1331–1345.