Personalized Real-Time Monitoring of Amateur Cyclists on Low-End Devices

Proof-of-Concept & Performance Evaluation

Mathias De Brouwer
IDLab – imec – Ghent University
mrdbrouw.DeBrouwer@UGent.be

Femke Ongenae
IDLab – imec – Ghent University
Femke.Ongenae@UGent.be

Glenn Daneels
IDLab – imec – University of Antwerp
Glenn.Daneels@UAntwerpen.be

Esteban Municio
IDLab – imec – University of Antwerp
Esteban.Municio@UAntwerpen.be

Jeroen Famaey
IDLab – imec – University of Antwerp
Jeroen.Famaey@UAntwerpen.be

Steven Latré
IDLab – imec – University of Antwerp
Steven.Latre@UAntwerpen.be

Filip De Turck
IDLab – imec – Ghent University
Filip.DeTurck@UGent.be

ABSTRACT

Enabling real-time collection and analysis of cyclist sensor data could allow amateur cyclists to continuously monitor themselves, receive personalized feedback on their performance, and communicate with each other during cycling events. Semantic Web technologies enable intelligent consolidation of all available context and sensor data. Stream reasoning techniques allow to perform advanced processing tasks by correlating the consolidated data to enable personalized and context-aware real-time feedback. In this paper, these technologies are leveraged and evaluated to design a Proof-of-Concept application of a personalized real-time feedback platform for amateur cyclists. Real-time feedback about the user's heart rate and heart rate training zones is given through a web application. The performance and scalability of the platform is evaluated on a Raspberry Pi. This shows the potential of the framework to be used in real-life cycling by small groups of amateur cyclists, who can only access low-end devices during events and training.

CCS CONCEPTS

• **Information systems** → **Data analytics**; *Data streams*; • **Theory of computation** → *Semantics and reasoning*;

KEYWORDS

stream reasoning; real-time feedback; cycling

ACM Reference Format:
Mathias De Brouwer, Femke Ongenae, Glenn Daneels, Esteban Municio, Jeroen Famaey, Steven Latré, and Filip De Turck. 2018. Personalized Real-Time Monitoring of Amateur Cyclists on Low-End Devices: Proof-of-Concept & Performance Evaluation. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3184558.3191648

1 INTRODUCTION

In recent years, the importance of using data to take strategic decisions in sports, both for professional events and amateur training, has significantly increased [18]. This monitoring is especially the case in cycling [5, 20]. Many cyclists are riding with a series of sensors measuring aspects such as heart rate, power, location, speed, altitude and cadence. However, existing cycling training apps, such as Strava[1], do almost all data reporting and analysis offline as a post-processing step.

Real-time collection and analysis can therefore bring important innovations in the cycling world, especially for amateur cyclists, who typically do not have the same resources as professional cycling teams. Enabling this can allow them to continuously monitor themselves, receive personalized feedback on how they are performing, and communicate with others during amateur cycling events, such as the Tour of Flanders for amateurs. To make the feedback more valuable for each rider, it should be personalized. Many different parameters define the physiological profile of a rider. For example, the resting and maximum heart rate define the training zone boundaries, which means other feedback may apply for different riders having the same heart rate. To be able to act immediately upon received feedback, its real-time aspect is important. Depending on the parameter, real-time requirements can differ. For example, power is much more oscillating than heart rate, meaning power feedback can only be considered real-time when updates occur at least every second, whereas for heart rate this can be up to 5 seconds.

To achieve the real-time data collection and analysis, two major technological innovations are researched. First, cycling events often take place at remote areas without any cellular connectivity. This is a challenge for data collection. Therefore, a novel IoT platform for challenging environments is described in this paper [8]. It serves as a mobile network layer with the bikes as nodes, allowing to send real-time sensor information from cyclist to cyclist, without the need of any Internet connection.

Second, a data analysis layer is researched, allowing for intelligent

[1]https://www.strava.com

real-time feedback before and during sports events, more specifically amateur cycling events. To make this feedback personalized and context-aware, the system should be able to take into account context information, such as rider profiles, route information, weather etc. Given all available context information and the observations of the different sensors, intelligent consolidation and analysis of this data is required. As this data is often heterogeneous, semantics are the ideal approach to tackle this issue [4]. Ontologies can be used to model the data and their relationships and properties. Stream reasoning techniques then allow to perform advanced processing tasks that enable to design personalized and context-aware real-time feedback, by mapping the continuous data streams on the available background knowledge modeled in an ontology [9]. For example, an incoming heart rate observation of a sensor can be linked to the corresponding rider, allowing to retrieve his profile information and determine his current heart rate training zone and associated feedback depending on this person's boundaries. An added advantage of semantics is the usage of generic queries, allowing context data, e.g., new sensor types, and queries to be added to a running system, without the need for adaptations.

Within the imec ICON project CONAMO (CONtinuous Athlete MOnitoring)[2], a Proof-of-Concept (PoC) application was realized, demonstrating what can be achieved by implementing a real-time feedback platform for cyclists using the two technological advances discussed above. To allow adoption by amateur cyclists, the real-time feedback system should be able to run on a low-end device, e.g., the GPS device or smartphone used on their bike. Therefore, the performance and scalability of the designed platform is evaluated on a Raspberry Pi.

The outline of this paper is as follows. Section 2 presents related work in the area of stream reasoning, semantics in sports, and IoT data collection. Section 3 details the PoC use case, while Section 4 presents its architecture set-up. The IoT platform for real-time data collection is described in Section 5, whereas Section 6 discusses the real-time feedback platform itself. In Section 7 and 8, the performance of the feedback system is evaluated on a Raspberry Pi. Finally, Section 9 and 10 discuss and conclude the main findings.

2 RELATED WORK

2.1 Stream reasoning

Data Stream Management Systems (DSMS) & Complex Event Processing (CEP) allow to query homogeneous streaming data structured according to a fixed data model [17]. In contrast to Semantic Web reasoners, DSMS & CEP are unable to deal with heterogeneous data sources and lack support for the integration of domain knowledge. To bridge this gap, stream reasoning focuses on the scalable and efficient adoption of Semantic Web technologies for streaming data [9]. In the past years, several RDF Stream Processing (RSP) engines have been developed [23], of which C-SPARQL [3] and CQELS [16] are the most well-known. They define a window on top of the stream and allow the registration of semantic queries which are continuously evaluated as data flows through the window. These RSP engines can thus filter & query a continuous flow of data, provide real-time answers to the registered queries and support the integration of domain knowledge into the querying process [9]. C-SPARQL enables RDFS reasoning, whereas CQELS does not include any reasoning support. Some preliminary research has been done on publishing RDF streams from low-end devices [22].

2.2 Semantic technologies in sports

Some preliminary research has been done about the adoption of semantic technologies for sports. The most advanced of these endeavors is the research conducted in context of the Lifewear ITEA Project [1, 21]. In this research, a wireless sensor network was designed to model sporters who are performing weight-lifting exercises inside a gymnasium. Semantic reasoning is employed to give feedback on their current training schema adherence and to generate alarms when they are taking their physical exercises to a dangerous level. Other endeavors are limited to the semantic annotation and retrieval of sports information [6, 26].

2.3 IoT platform for challenging environments

Traditional IoT platforms are based on wireless technologies that are highly dependent on a dense infrastructure of interconnected base stations, and are therefore not suitable for rural, remote and challenging areas with low population density [12, 14, 15]. To extend the IoT application to these environments, a number of solutions have been proposed [2, 7, 19], using the traditional mobile networks (e.g., GPRS, LTE), NB-IoT [11], LoRa, or SIGFOX to interconnect the IoT devices with a high-speed back-haul network. These infrastructure-based approaches have the disadvantage that deploying and maintaining base stations is expensive and the IoT devices can only function in the presence of a base station. Infrastructure-less approaches on the other hand [8, 10, 25] can work locally without the need to be connected to the Internet. They are based in low-power multi-hop Wireless Sensor Network (WSN) technologies such as DASH7, Zigbee, and 6TiSCH.

3 USE CASE SCENARIO

The goal is to give real-time personalized feedback to amateur cyclists on their heart rate and heart rate training zone they are currently in. This helps riders to perform the most efficient training instead of over or under training. Important is the personalized approach: all feedback should be adapted to the profile of the rider.

For cyclists, seven heart rate training zones can be distinguished[3]: (1) recovery, (2) long slow distance (LSD), (3) extensive endurance, (4) tempo endurance, (5) block training, (6) extensive interval, (7) anaerobic. For each individual, the boundaries between these training zones can be different. To be able to personalize the feedback, it is therefore important to determine these boundaries based on some profile information.

Based on the resting heart rate and maximum heart rate of a person, the upper bounds of the different training zones can be determined by applying the Karvonen formula [13]:

$$UB_{tz} = intensity_{tz} \times (HR_{max} - HR_{rest}) + HR_{rest} \tag{1}$$

The intensity values for the different training zones are 0.60, 0.64, 0.70, 0.78, 0.84, 0.89 and 1.00 respectively. To determine the resting and maximum heart rate of a person, one should ideally execute

[2]https://www.imec-int.com/nl/imec-icon/research-portfolio/conamo

[3]Provided by Energy Lab, partner of the CONAMO project.

professional lab tests, which are not always feasible or realistic for amateur cyclists. Therefore, expertise rules of thumb exist, allowing to determine realistic default values from profile parameters:

- Resting heart rate: this heart rate depends on the level of sportiveness of the person. For a low level of sportiveness, this value is around 65. Similarly, this value is 55 for a medium level and 50 for a high level.
- Maximum heart rate: this value typically decreases with age, and can be estimated as $220 - age$.

By using these expertise rules and the Karvonen formula, a rider's heart rate training zone boundaries can be determined by solely requesting the person's age and level of sportiveness. If a person knows his own maximum and/or resting heart rate himself, these values can of course be used instead of the estimated ones.

4 ARCHITECTURE SET-UP

Figure 1 visualizes the PoC set-up with two riders[4]. It consists of two parts: the IoT platform for challenging environments (see Section 5) and the real-time feedback system (see Section 6).

For personalization purposes, a mobile Android app has been developed, requesting the rider to fill in some profile information before starting the PoC. An Android smartphone running this app is connected to each bike. For reasons explained in Section 3, date of birth, level of sportiveness and, if known, resting heart rate and maximum heart rate are requested. Name & gender are asked to allow unique identification and later comparison with gender-specific benchmarks. All profile information is stored as context data in the feedback system, to avoid that it needs to be entered every time. Screenshots of the app questions are shown in Figure 2.

Each rider is wearing a heart rate sensor, measuring the heart rate approximately every second. These observations are sent over different stages and technologies to the real-time feedback system in the back-end, where a stream reasoning framework is deployed. The real-time feedback is shown to the user on a screen, which can for example be on a GPS device or smartphone on the bike or installed in the car, using a locally running web application.

The feedback system is installed on a Raspberry Pi 3, Model B. This low-end device is chosen because it acts as a simple Linux computer, allowing any system running on a laptop to be transported easily. This is for example not possible on an Android tablet.

In Figure 3, a screenshot of the web application for the real-time feedback system for one rider is shown. Feedback is given about the heart rate, corresponding training zone, and relative training zone distribution for the duration of the ride. When the rider stops cycling, the real-time feedback can be stopped, showing a final training zone distribution with feedback about all training zones. Visual colored feedback (green, orange or red) is given to indicate the intensity of the current effort in relation to a person's aerobic and anaerobic threshold.

5 IOT PLATFORM FOR CHALLENGING ENVIRONMENTS

To support real-time collection of sensor data, an IoT platform is proposed that can reliably disseminate data in various dynamic

[4]It should be noted that the system works with an arbitrary number of riders.

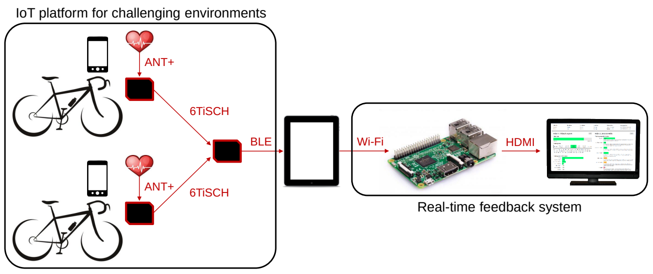
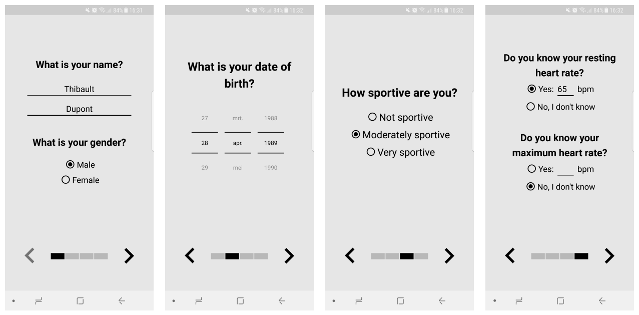

IoT platform for challenging environments
ANT+
6TiSCH
BLE
ANT+
6TiSCH
Wi-Fi
HDMI
Real-time feedback system

Figure 1: PoC set-up



What is your name?
Thibault
Dupont
What is your gender?
[✓ Male] Female

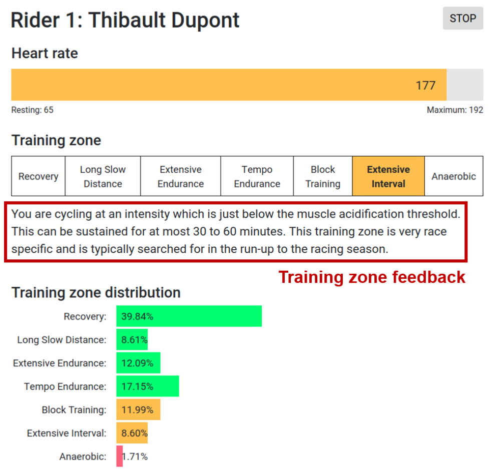
What is your date of birth?
27 mrt. 1988
28 apr. 1989
29 mei 1990

How sportive are you?
Not sportive
[✓ Moderately sportive]
Very sportive

Do you know your resting heart rate?
[✓ Yes: 65 bpm]
No, I don't know

Do you know your maximum heart rate?
Yes: ___ bpm
[✓ No, I don't know]

Figure 2: Screenshots of the Android app requesting a rider's profile information, for feedback personalization



Rider 1: Thibault Dupont

STOP

Heart rate

177

Resting: 65 Maximum: 192

Training zone

Recovery	Long Slow Distance	Extensive Endurance	Tempo Endurance	Block Training	**Extensive Interval**	Anaerobic

You are cycling at an intensity which is just below the muscle acidification threshold. This can be sustained for at most 30 to 60 minutes. This training zone is very race specific and is typically searched for in the run-up to the racing season.

Training zone feedback

Training zone distribution

Recovery:	39.84%
Long Slow Distance:	8.61%
Extensive Endurance:	12.09%
Tempo Endurance:	17.15%
Block Training:	11.99%
Extensive Interval:	8.60%
Anaerobic:	1.71%

Figure 3: Screenshot of the web app for the real-time feedback system, shown for only one rider

and challenging environments [8]. This platform is shown in the left side of Figure 1. It is a hybrid solution that combines the advantages of infrastructure-based and infrastructure-less low-power connectivity for IoT in a single multi-modal platform. It uses a variety of Low-power Wireless Personal Area Network (LoWPAN) technologies (e.g., BLE and ANT+) to connect to different sensors that monitor the entity (i.e., the rider), while it also employs an ad-hoc, long-range and multi-hop WSN that reliably disseminates the monitored data to the network sink in real-time. The proposed multi-modal platform provides a highly versatile IoT solution for challenging environments compared to state of the art solutions, with infrastructure-less data dissemination, sporadic disruption-

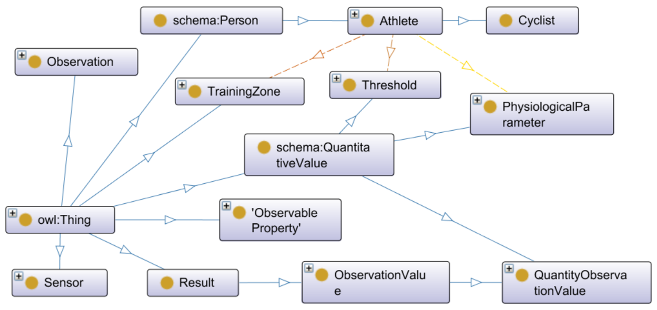


Figure 4: Core structure of the cycling ontology[5]

tolerant Internet uplink, and support for heterogeneous wireless sensors and actuators. To allow comprehension of the overall PoC, a short description of the device functionality and data dissemination process of the platform is provided in this section. More details can be found in Daneels et al. [8].

5.1 Device functionality

A platform node will collect data from the riders using the sensors it connects to, i.e., it connects to the heart rate monitor of the rider as shown in Figure 1. The captured data is periodically forwarded into the network to reach the network sink, where it can be analyzed. Being part of a multi-hop network, each platform node is also responsible for receiving incoming data from other nodes, and relaying those data further upwards to the sink.

The platform sink serves as a collector of the monitored sensor data coming from the platform nodes. The sink itself does not connect to any sensor device, but uses BLE to connect to other peripherals in order to analyze and/or visualize the incoming data. As shown in Figure 1, it sends the monitored data to a tablet which is connected to the proposed real-time feedback system.

5.2 Data dissemination

The IoT platform supports end-to-end dissemination of real-time data from sensors to the sink and in the reverse direction over a range up to several kilometers. Rather than using external infrastructure (e.g., 4G or LPWAN), it employs a multi-hop communication network using the IEEE 802.15.4g physical layer in combination with IEEE 802.15.4e MAC layer. The IEEE 802.15.4g amendment was recently added to the IEEE 802.15.4 standard, supports an infrastructure-less mode, and is tailored for low-power, long-range communication using the 868 MHz frequency band. Communication at this sub-1GHz frequency band allows for a connection between two nodes up to several hundreds of meters. The IEEE 802.15.4g physical layer is combined with the 6TiSCH architecture that combines industrial performance in terms of reliability and power consumption with a full IPv6-enabled IoT upper stack [24]. The key component of the 6TiSCH stack is the TSCH mode of the IEEE 802.15.4e MAC layer that combines channel hopping to avoid external interference and multi-path fading with a TDMA-based TX/RX schedule. This results in an extremely reliable and energy-efficient multi-hop IoT network, that can disseminate data in real-time over a distance of several kilometers. In the PoC set-up shown in Figure 1, there was a direct 6TiSCH connection between the nodes and the sink.

Listing 1: Example of a heart rate sensor observation, described in JSON-LD

```
{
  "@context": {
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "schema": "http://schema.org/",
    "sosa": "http://www.w3.org/ns/sosa/",
    "madeBySensor": { "@id": "sosa:madeBySensor", "@type": "@id" },
    "resultTime": { "@id": "sosa:resultTime", "@type": "xsd:dateTime" },
    "hasResult": { "@id": "sosa:hasResult" },
    "value": { "@id": "schema:value", "@type": "xsd:float" },
    "unitText": { "@id": "schema:unitText", "@type": "xsd:string" }
  },
  "@id": "http://idlab.ugent.be/conamo/sensors/
          Observation_HeartRateSensor_1593_2017_09_16_21_33_25_850Z",
  "@type": "http://www.w3.org/ns/sosa/HeartRateObservation",
  "madeBySensor": "http://idlab.ugent.be/conamo/sensors#HeartRateSensor_1593",
  "resultTime": "2017-09-16T21:33:25.850Z",
  "hasResult": {
    "@id": "http://idlab.ugent.be/conamo/sensors/
          ObservationValueSensor_HeartRateSensor_1593_2017_09_16_21_33_25_850Z",
    "@type": "http://idlab.ugent.be/conamo-vocab/sosa#HeartRateObservationValue",
    "value": "137", "unitText": "bmp"
  }
}
```

6 REAL-TIME FEEDBACK SYSTEM

The main part of the feedback system is a stream reasoning framework running on a low-end device. To enable visual feedback to the user, a front-end application, such as the locally running web application shown in Figure 3, should be implemented. Such an application can be viewed on the low-end device itself, or on any other device connecting to the same network.

6.1 Ontology

A cycling ontology was developed[6], which is based on the SOSA ontology[7] (Sensor, Observation, Sample and Actuator). An overview of the core structure of the ontology is shown in Figure 4. The ontology allows to define a cyclist, a training zone and the physiological profile of the cyclist, including training zone boundaries. Each training zone is annotated with comments containing feedback compiled by domain experts. Furthermore, sensor observations can be defined. Support is provided for location and quantity observation values. The latter include all relevant quantity observations with a value & unit, e.g., heart rate, altitude, power and speed of a cyclist. This generic definition enables to define corresponding generic reusable queries, as is explained in Section 6.3.

The ontology is used to construct the knowledge base used by the RSP engine. Hence, queries and data (i.e., both static context data and input stream data), all need to be modeled in this ontology. Listing 1 shows an example of a heart rate sensor observation as it is posted on the stream, described in JSON-LD.

6.2 Usage of C-SPARQL

The RSP engine used in the system is C-SPARQL, because of its supports for static context data, which CQELS has not. In C-SPARQL, a registered continuous query is executed when the corresponding window is triggered, i.e., after the slide.

To be able to work with a RESTful interface, the RSP Service Interface for C-SPARQL[8] is used. Upon start-up, a WebSocket server

[5]Created with the Protégé ontology editor (https://protege.stanford.edu).
[6]Publicly available at https://github.com/IBCNServices/cyclists-monitoring.
[7]http://www.w3.org/ns/sosa
[8]https://github.com/streamreasoning/rsp-services-csparql

Listing 2: `getQuantityObservationValue` query

```
REGISTER STREAM getQuantityObservationValue AS

PREFIX f: <http://larkc.eu/csparql/sparql/jena/ext#>
PREFIX schema: <http://schema.org/>
PREFIX sosa: <http://www.w3.org/ns/sosa/>
PREFIX conamo-profile: <http://idlab.ugent.be/conamo-vocab/profile#>
PREFIX conamo-sosa: <http://idlab.ugent.be/conamo-vocab/sosa#>

SELECT
  ?deviceUUID ?firstName ?lastName ?gender ?birthDate ?sensor ?o ?time ?unit ?value
FROM STREAM <http://idlab.ugent.be/conamo/stream> [RANGE 10s STEP 1s]
FROM <http://localhost:8080/conamo/context/conamo-sensors.rdf>
FROM <http://localhost:8080/conamo/context/conamo-riders.rdf>
WHERE {
  ?o sosa:hasResult ?ov . ?o sosa:resultTime ?time . ?sensor sosa:isHostedBy ?platform .
  ?platform conamo-sosa:UUID ?deviceUUID . ?rider conamo-profile:monitoredBy ?platform .
  ?rider schema:givenName ?firstName . ?rider schema:familyName ?lastName .
  ?rider schema:gender ?gender . ?rider schema:birthDate ?birthDate .
  ?ov a conamo-sosa:QuantityObservationValue .
  ?ov schema:unitText ?unit . ?ov schema:value ?value .
  { SELECT ?sensor ( MAX ( f:timestamp (?x, sosa:madeBySensor, ?sensor) ) AS ?ts ) WHERE { ?x
          sosa:madeBySensor ?sensor . } GROUP BY ?sensor }
  FILTER ( f:timestamp (?o, sosa:madeBySensor, ?sensor) = ?ts )
}
```

is started. For each registered query, the query output stream with the variable bindings is published on a query-specific WebSocket URL. Any interested client service, e.g., a web application, can then open a WebSocket connection to the relevant URL, where the variable bindings can then be received in real-time as messages.

6.3 C-SPARQL queries

To enable real-time feedback, C-SPARQL queries can be defined & registered in the engine. Two queries were constructed that allow feedback on a rider's heart rate and corresponding training zone.

The `getQuantityObservationValue` query (Listing 2) analyzes the data to see if there is any `QuantityObservationValue`, e.g., the subclass `HeartRateObservationValue`, in the stream that can be linked to the rider's profile. Any observation of a sensor that produces a value with a unit as result, and is associated to an existing rider profile, is filtered by the query. For each sensor, only the most recent observation is outputted. The generic structure of the query exploits the generic structure of the cycling ontology explained in Section 6.1, as it aggregates results of all sensors producing `QuantityObservationValues`.

The `getTrainingZone` query (Listing 3) derives the current heart rate zone of the riders. For any `HeartRateObservationValue` in the input window, the query tries to link this observation to the corresponding rider's profile, to select the heart rate training zone corresponding to the heart rate value. The associated training zone feedback is given in the result as well, as defined in the cycling ontology. For each heart rate sensor known by the system, only the most recent heart rate sensor observation is considered.

Both queries have the same window parameters. This is required for the real-time feedback to be consistent: if the heart rate value of a rider changes, his training zone changes accordingly.

6.4 Dynamic approach

The advantage of the described framework is its dynamic aspect. To create personalized feedback, only the context information needs to be updated while the feedback system is running. For example, assume a new rider is joining the group of amateur cyclists. To be able to also output query results concerning this person's heart rate and

Listing 3: `getTrainingZone` query

```
REGISTER STREAM getTrainingZone AS

PREFIX f: <http://larkc.eu/csparql/sparql/jena/ext#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
PREFIX sosa: <http://www.w3.org/ns/sosa/>
PREFIX conamo-profile: <http://idlab.ugent.be/conamo-vocab/profile#>
PREFIX conamo-sosa: <http://idlab.ugent.be/conamo-vocab/sosa#>

SELECT
  ?deviceUUID ?firstName ?lastName ?gender ?birthDate ?sensor ?o ?time ?tzName ?tzDescription
  ?tzLB ?tzUB
FROM STREAM <http://idlab.ugent.be/conamo/stream> [RANGE 10s STEP 1s]
FROM <http://localhost:8080/conamo/context/conamo-sensors.rdf>
FROM <http://localhost:8080/conamo/context/conamo-riders.rdf>
WHERE {
  ?o sosa:hasResult ?ov . ?o sosa:resultTime ?time . ?sensor sosa:isHostedBy ?platform .
  ?platform conamo-sosa:UUID ?deviceUUID . ?rider conamo-profile:monitoredBy ?platform .
  ?rider schema:givenName ?firstName . ?rider schema:familyName ?lastName .
  ?rider schema:gender ?gender. ?rider schema:birthDate ?birthDate .
  ?ov a conamo-sosa:HeartRateObservationValue . ?ov schema:value ?value .
  ?rider conamo-profile:hasTrainingZone ?tzRider . ?tzRider conamo-profile:hasLowerBound ?tzLB .
  ?tzRider conamo-profile:hasUpperBound ?tzUB . ?tzRider a ?tz . ?tz rdfs:label ?tzName .
  ?tz rdfs:comment ?tzDescription . ?tz rdfs:subClassOf conamo-profile:TrainingZone .
  { SELECT ?sensor ( MAX ( f:timestamp (?x, sosa:madeBySensor, ?sensor) ) AS ?ts ) WHERE { ?x
          sosa:madeBySensor ?sensor . } GROUP BY ?sensor }
  FILTER ( f:timestamp (?o, sosa:madeBySensor, ?sensor) = ?ts )
  FILTER ( ?value >= ?tzLB ) FILTER ( ?value <= ?tzUB )
}
```

training zone, the static context data used by C-SPARQL only needs to be updated by adding the sensor and profile information related to this rider and his bike. Besides, no runtime changes are needed to the C-SPARQL back-end server itself and the registered C-SPARQL queries. If a front-end application is able to deal with an arbitrary amount of riders, this application can simply continue running as before; the new bindings related to the new rider will also be sent as messages to the relevant WebSocket URLs. Similarly, new sensor types, new training zones or additional feedback can all be added to the context data, i.e., ontology, without having to change anything to the implemented platform. Also, for new queries, the query will be continuously executed once registered, sending its results as messages on its new WebSocket URL. Summarized, context data and queries can all be added, removed or updated at any time while running the system, without the need of other adaptations.

7 EVALUATION SET-UP

It is important that the real-time feedback system, described in Section 6, is capable of handling a group of amateur cyclists. However, more riders means more sensor observations and more static context data for a C-SPARQL query to work with. As low-end devices are typically characterized by limited memory and limited CPU capabilities, tests have been performed to assess how many riders can be supported by one Raspberry Pi. Furthermore, the impact of the heart rate frequency and C-SPARQL query parameters on the performance has been evaluated as well. The tests have been performed on the use case described in Section 3, with the set-up described in Section 4, and with the stream reasoning framework and queries as given in Section 6.

7.1 Hardware specifications

The used low-end device for the tests described in this section is a Raspberry Pi 3, Model B. This device has a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU, 1GB RAM and MicroSD storage[9].

[9]https://www.raspberrypi.org/products/raspberry-pi-3-model-b

The operating system is Raspbian Stretch with desktop, version of November 2017, with kernel version 4.9[10]. The desktop version is chosen over the minimal OS, providing the ability to run and show a front-end application on the same device. This application may then consume the C-SPARQL query results to visualize the real-time feedback. When no foreground app is running on the Raspberry Pi, approximately 900MB of the memory is unused, and the active CPU usage is not higher than 1%.

7.2 Evaluation parameters

First, not all available system memory may be in use when the feedback system is running. If that happens, memory issues may arise, disallowing the system to run as it should. Hence, the total memory usage should be lower than the available memory.

Second, the feedback should remain real-time. In concrete, this means that the system cannot start lagging, i.e., the total query execution time may not surpass the window sliding step. If that happens, the same query is triggered to execute again, while the previous execution is still ongoing.

Third, feedback should be updated real-time for each rider. This means that if the static context data describes n rider profiles, each query output should preferably contain n results, i.e., 1 for each rider[11], as it is guaranteed that only the most recent observation of each sensor is filtered by both queries. Hence, if this amount is smaller than n, no observation is present in the window for at least one rider. This is an indication that C-SPARQL is not able to run as it should, because it needs more resources than available.

7.3 Test scenarios

To perform the evaluation, a simulation of a real-life set-up has been performed. Different scenarios have been simulated. First, the *number of riders in the system* has been evaluated. With respect to the value of the *sliding step* of both C-SPARQL queries, the distinction between the ideal and extreme case has been made. In the ideal case, the queries are executed every second, to obtain an output event rate of approximately 1 second. This would then allow any front-end application to be updated every second, which is desirable for a real-time system. However, in the extreme case, one must consider the largest sliding step at which the updates can still be regarded as real-time. For heart rate values, this value is approximately 5 seconds, as a person's heart rate curve is typically relatively smooth. If the interval between updates is more than 5 seconds, this is no longer acceptable[12]. Besides the query sliding step, the impact of the *query window size* and the *event rate*, i.e., the time between successive incoming heart rate observations per rider, has also been investigated.

For each simulation, queries and context data were specified according to the query parameters and number of riders. Each time, a C-SPARQL[13] server has been started, and for each rider, a parallel script has then been launched to post randomly generated heart rate observations on the registered RDF stream at the specified event rate. In this way, the arrival pattern of heart rate observations of different sensors on the input stream is realistic: arrivals of the same period are close to each other in time, but their arrival order is undefined and can vary. To obtain comparable results, the amount of heart rate observations was chosen as such that both queries were executed at least 50 times during each simulation.

The measurements used for evaluation are taken from the C-SPARQL measurements provided in the generated log files. Based on the performance evaluation parameters defined in Section 7.2, the following metrics were calculated for each simulation from the log output: (1) average query execution time[14], (2) 90th percentile of query execution times, (3) maximum query execution time, (4) average memory usage, (5) maximum memory usage, (6) normalized average number of query results. This last parameter is defined as the average number of output bindings per query execution, divided by the number of rider profiles. As at most one result per rider can be present, this number should always be 1 or smaller[15]. The closer this number is to 1, i.e., the closer the amount of results is to the number of riders, the better, as explained in Section 7.2.

8 RESULTS

In Figure 5, the results of the evaluation of the number of riders in the system are shown, for a fixed event rate and query window size. In the plots, query execution times (average, 90th percentile, maximum) are displayed, as well as the normalized average number of query results. Memory usage is omitted from the plots, as these values never exceed 50MB. Results are compared between the ideal and the extreme case (query sliding step of 1 vs. 5 seconds). The plots show the simulation results for 1 to 20 riders.

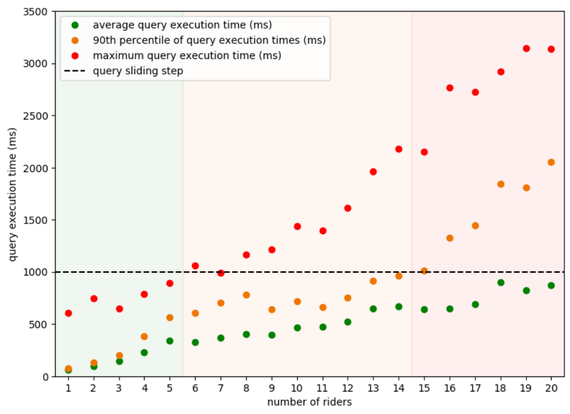
As observed in Figure 5a and 5b, the average query execution times never surpass the query sliding step. However, the maximum execution time does exceed this value, respectively from 6 riders and 13 riders onwards for the ideal and extreme case. The 90th percentile rises above this value from 15 and 19 riders on respectively. Based on these two events, the charts have been divided into three zones (green, orange, red), which are discussed in Section 9. Associated with the rising execution times, the average number of query results decreases from 6 riders and onwards in the ideal case (Figure 5c), with a significant drop for 15 to 20 riders. In the extreme case (Figure 5d), this value only deviates from 1 for 15 riders and more.

The impact of the query window size on the calculated metrics is not plotted, as results show this impact is rather small. For a query sliding step of 1 second, 1 second between heart rate observations, and 5 rider profiles[16], the maximum query execution times are not impacted for a query window size increasing from 1 to 20. The average increases from 95 to 324 seconds. The sliding step of 1 second is never exceeded by the 90th percentile, and not by more than 100ms by the maximum value. Memory usage does not exceed 20MB. Moreover, the number of results is not impacted at all.

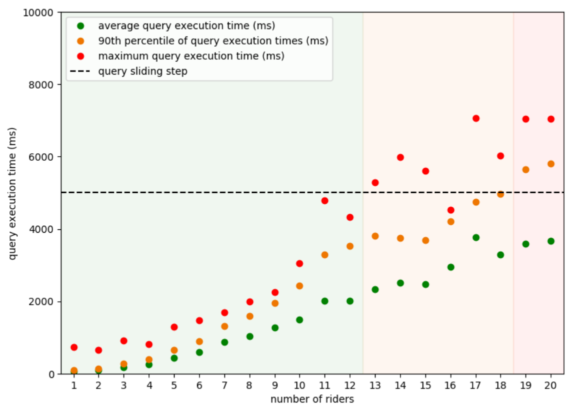
Finally, the results of evaluating the impact of the event rate are shown in Figure 6. With a fixed query window size of 10 seconds,

[10]https://www.raspberrypi.org/downloads/raspbian
[11]Assuming only active riders are described in the static data.
[12]According to the expertise of Energy Lab, partner of the CONAMO project.
[13]C-SPARQL version 0.9.7 (https://github.com/streamreasoning/CSPARQL-engine)

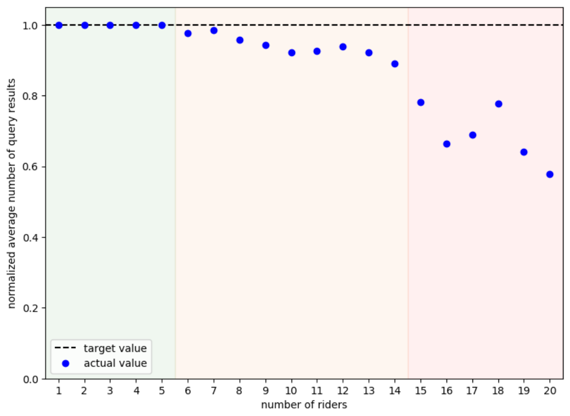
[14]As both queries are executed in parallel and on their own, the individual execution times are considered.
[15]When the load on C-SPARQL is too high, runtime exceptions can exceptionally lead to more results than normal. As these results cannot be used, these values are omitted when constructing the charts, to avoid misinterpretations.
[16]The amount of riders was set to 5 because this is the highest number that is still in the green zone on the test results for a sliding step of 1 second in Figure 5.



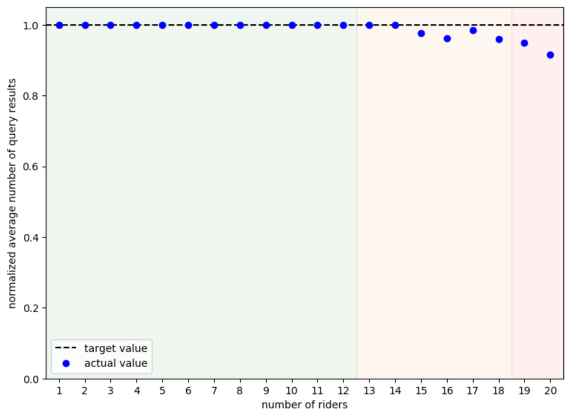
(a) Query execution times for a sliding step of 1 second (ideal case)



(b) Query execution times for a sliding step of 5 seconds (extreme case).



(c) Normalized average number of query results for a sliding step of 1 second (ideal case)



(d) Normalized average number of query results for a sliding step of 5 seconds (extreme case).

Figure 5: Query execution times and normalized average number of query results for 1 to 20 rider profiles, measured on a Raspberry Pi 3, Model B. Time between incoming heart rates is 1 second for each rider. Query window size is 10 seconds.

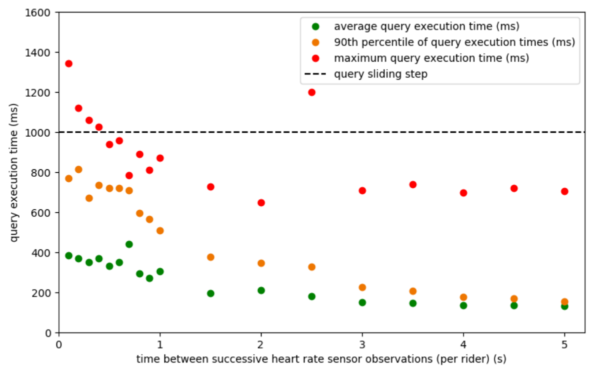


Figure 6: Query execution times for varying time between successive heart rate observations per rider, for 5 riders. Query window size is 10 seconds, sliding step is 1 second.

sliding step of 1 second and 5 rider profiles[16], too high execution times only occur when the time between successive heart rate sensor observations gets 200ms and smaller. The 90th percentile however already starts increasing when this time gets smaller than 1 second (i.e., the sliding step), showing that high execution times start occurring more and more. Memory usage is not an issue, as it does not exceed 35MB. The amount of results is rot really impacted.

9 DISCUSSION

The results of the performance evaluation described in Section 8 show the bottlenecks of the Raspberry Pi as low-end device to run the stream reasoning framework of the real-time feedback system. Recalling the performance evaluation parameters discussed in Section 7.2, it is clear that memory usage is not an issue for the use case described in Section 3. The main performance bottlenecks are the query execution times exceeding the sliding step, causing performance issues resulting in fewer query results than expected.

As the green zone in the left of Figure 5 shows, up to 5 riders can be supported without any problem in the ideal case of one query execution per second. In the orange zone, at least 90% of the query executions are finished within 1 second[17]. Issues occur but not yet constantly, and query executions do not always have results for each rider. Therefore, it is no longer ideal to use a Raspberry Pi, although the feedback may still be acceptably real-time at most moments. In the red zone however, high query execution times occur much more frequently, heavily impacting the amount of query results. As real-time feedback cannot be guaranteed any longer, it is not acceptable to use the system for this amount of riders.

Similarly, the same observations can be made in the extreme case of only one update every 5 seconds. However, compared to the ideal case, a group of 12 instead of 5 riders is still in the green zone, and 18 instead of 14 riders are still in the orange zone.

Figure 6 shows that higher event rates lead to higher restrictions, especially when having more than one heart rate observation per rider per second. However, the rate of 1 observation per second, used for the evaluation in Figure 5, is realistic for the set-up with the IoT platform described in Section 5. Hence, it can be stated that one Raspberry Pi can support a group of 5 to 12 amateur cyclists to provide real-time feedback on their heart rate and training zones. This is acceptable in the given use case, as most groups do not consist of more than 12 people. Otherwise, it is always possible to parallelize the processing on more than one Raspberry Pi.

To perform a user evaluation, the use case scenario described in Section 3 has been implemented for an end-to-end demo with two riders at the Media Fast Forward event[18] in December 2017. During the demo given, the users were generally interested in the real-time feedback given on the screen. Most user feedback received highlighted the added value of the personalization aspect. Some suggestive comments were made as well, mostly about the lack of integration of other sensors. Many people seemed to be interested in more extended feedback about other sensor data, i.e., not only limited to heart rate training zones. Also the integration of other context data such as route and weather information would be of interest when the system is applied for real-life cycling. This is easily possible by the use of semantics. In general, the generic ontology described in Section 6.1 in combination with reasoning, allows to make any use case to be made context-aware and personalized to the desired extent.

10 CONCLUSION

In this paper, a PoC of a personalized real-time feedback platform for amateur cyclists on low-end devices has been presented. By discussing a use case for heart rate and training zone feedback, it is explained how semantics and stream reasoning are applied to support the design of personalized and context-aware real-time services. In this way, amateur cyclists can monitor themselves, receive personalized feedback on how they are performing, and communicate with others during amateur cycling events. This is currently not yet possible with existing cycling training apps like Strava. A performance evaluation on a Raspberry Pi has shown that the system can give real-time heart rate and training zone feedback every 1 to 5 seconds for groups of up to 12 amateur cyclists. This demonstrates the potential of the framework to be used in real-life amateur cycling. Future work consists of adopting the system for extended use cases, addressing more than only the heart rate, e.g., power, location, speed, altitude, and taking into account more context information, e.g., the rider's power profile, the gradient at different route segments, weather information etc. This would illustrate the benefits of using semantics and reasoning even more.

[17] Typically, in the orange zone, the maximum value occurs in the first query execution because of the materialization process, and other executions take much less time. This is no longer the case in the red zone.

[18] Organized annually in Brussels, Belgium by VRT (https://www.mediafastforward.be).

ACKNOWLEDGMENTS

F. Ongenae is funded by BOF postdoc grant from UGent. Part of this research was funded by the FWO SBO S004017N IDEAL-IoT, the FWO SBO 150038 DiSSeCt, and the imec ICON CONAMO. CONAMO is funded by imec, VLAIO, Rombit, Energy Lab and VRT. Special thanks to Energy Lab for providing the domain expertise.

REFERENCES

[1] ITEA Office Association. 2013. The Lifewear ITEA Project: Mobilized Lifestyle With Wearables. (2013). Retrieved December 21, 2017 from https://itea3.org/project/lifewear.html

[2] S. Aust et al. 2011. Sub 1GHz wireless LAN deployment scenarios and design implications in rural areas. In *GLOBECOM 2011*. IEEE, 1045–1049.

[3] D. F. Barbieri et al. 2010. C-SPARQL: a continuous query language for RDF data streams. *International Journal of Semantic Computing* 4, 1 (2010), 3–25.

[4] P. Barnaghi et al. 2012. Semantics for the Internet of Things: early progress and back to the future. *International Journal on Semantic Web and Information Systems* 8, 1 (2012), 1–21.

[5] R. Beecham et al. 2014. Exploring gendered cycling behaviours within a large-scale behavioural data-set. *Transportation Planning and Technology* 37, 1 (2014), 83–97.

[6] F. Camous et al. 2008. Capturing personal health data from wearable sensors. In *SAINT 2008*. IEEE, 153–156.

[7] M. Centenaro et al. 2016. Long-range communications in unlicensed bands: The rising stars in the IoT and smart city scenarios. *IEEE Wireless Communications* 23, 5 (2016), 60–67.

[8] G. Daneels et al. 2017. Real-time Data Dissemination and Analytics Platform for Challenging IoT Environments. In *GIIS 2017*. IEEE.

[9] D. Dell'Aglio et al. 2017. Stream reasoning: A survey and outlook. *Data Science* Preprint (2017), 1–25.

[10] I. Foche-Pérez et al. 2016. A dual IEEE 802.11 and IEEE 802.15–4 network architecture for energy-efficient communications with low-demanding applications. *Ad Hoc Networks* 37 (2016), 337–353.

[11] J. Gozalvez. 2016. New 3GPP Standard for IoT [Mobile Radio]. *IEEE Vehicular Technology Magazine* 11, 1 (2016), 14–20.

[12] J. Jin et al. 2014. An Information Framework for Creating a Smart City Through Internet of Things. *IEEE Internet of Things Journal* 1, 2 (2014), 112–121.

[13] M. Kent. 2006. *Oxford dictionary of sports science and medicine*. Vol. 10. Oxford university press.

[14] J. Lanza et al. 2016. Smart City Services over a Future Internet Platform Based on Internet of Things and Cloud: The Smart Parking Case. *Energies* 9, 9 (2016), 719.

[15] S. Latre et al. 2016. City of things: An integrated and multi-technology testbed for IoT smart city experiments. In *ISC2 2016*. IEEE, 1–8.

[16] D. Le-Phuoc et al. 2011. A native and adaptive approach for unified processing of linked streams and linked data. In *ISWC 2011*. Springer, 370–388.

[17] A. Margara et al. 2014. Streaming the web: Reasoning over dynamic data. *Journal of Web Semantics* 25 (2014), 24–44.

[18] P. O'Donoghue et al. 2014. *Data analysis in sport*. Routledge.

[19] C. Pham et al. 2016. Low-cost, Long-range open IoT for smarter rural African villages. In *ISC2 2016*. IEEE, 1–6.

[20] G. Romanillos et al. 2016. Big data and cycling. *Transport Reviews* 36, 1 (2016), 114–133.

[21] G. Rubio Cifuentes et al. 2012. A novel context ontology to facilitate interoperation of semantic services in environments with wearable devices. In *OTM 2012*. Springer, 495–504.

[22] Y. A. Sedira et al. 2017. MobileWave: Publishing RDF Streams From SmartPhones. In *ISWC 2017*. Springer.

[23] X. Su et al. 2016. Stream reasoning for the Internet of Things: Challenges and gap analysis. In *WIMS 2016*. ACM, 1–10.

[24] X. Vilajosana et al. 2017. *Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration*. Technical Report RFC8180. IETF.

[25] Z. Zaidi et al. 2017. Wireless multihop backhauls for rural areas: A preliminary study. *PLoS ONE* 12, 4 (2017), e0175358.

[26] J. Zhai et al. 2010. Semantic retrieval for sports information based on ontology and SPARQL. In *ISME 2010*. IEEE, 395–398.