Real-time location-based services for running races

Swen-Peter Ekkebus, Maarten M. Fokkinga, Nirvana Meratnia
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente, Enschede - The Netherlands
http://wwwhome.cs.utwente.nl/~ekkebus/
[Ekkebus, Fokkinga, Meratnia]@cs.utwente.nl

Abstract. This paper discusses the requirements and a basic system architecture for a real-time location-based service (LBS) for long-term speed running races. The envisioned system tracks the participating runners of the race and from this information spectators receive different types of services related to the race. The paper presents a design for a system, which combines GPS, cellular-network services and the Internet.

1 Introduction

Location-based services (LBS) have become an important part of our daily lives as a source of information; they can be accessed from almost anywhere using various wireless networks. Thanks to satellite positioning systems, recent LBSs offer more valuable services and are easier to use.

The combination of LBS, wireless networks and satellite positioning systems offer many opportunities to develop new services. This paper discusses the requirements and a basic architecture for real-time location-based services for long-term speed running races; it is based on lessons learnt form our attempt to develop a real time tracking system [1], which was first applied in a long-term relay-race. This system tracks the position of the participating runners in 'real-time' using GPS and transfers the runners' position using GSM services. It uses GIS utilities to make the information available through a map on a website.

The primary goal of this research is to add LBS services on top of this tracking system in order to give further insight in the race and to provide useful services to the spectators of the race.

2 Existing systems

In the world of sport, especially running races, there are just a few public known and automated systems, which give a real-time overview of the race and its participants. Existing systems in the running race domain can be categorized into three categories:

1. *Sensor-based systems:* these systems use a chip that is attached to the runner. The chip is censored by receivers that are located at fixed points along the racing track. (Example: ChampionChip [L2].)
2. *Measurement-based systems:* these systems also use a chip that is attached to the runner; it calculates the runner's speed and travelled distance. Subsequently, this information is sent to a central station to calculate the position of the runner on the track. (Example: Speedometer [L3].)
3. *Tracking-systems:* these systems use satellite-positioning techniques to track the runner's position. These positions are then sent to a central station and the runner positions are projected on the map. (Example: BPS [1][L1], Competitio [L4].)

The primary goal of sensor-based and measurement-based systems is timing. These systems can only be applied in races with a static racing route (marathons etc.), since they do not truly track the runners. Measurement-based systems need synchronisation in order to provide accurate information, since not all runners walk the 'shortest' path. Only the tracking systems are able to track races without predefined racing routes. They do not need any synchronisation because the position updates are independent of each other. Our tracking system, BPS [1][L1] was first applied in a large relay-race and was built from off-the-shelve building blocks (GPS, GSM, a website). The runner's positions were updated every 15 seconds.

The purpose of all these systems is to provide real-time information about the positions of the runners in the race and in some cases make the race more spectators friendly. Most of these systems do not offer any LBS; some systems can only be watched locally at the finish line and others only offer a web-site with limited user interaction.

The system proposed in this paper is a tracking system, providing real-time (interactive) location-based services to the end-user.

3 System requirements and architecture

To be able to provide a service as described above, the following requirements should be fulfilled by the system that provides the service. The requirements are ordered according to the five distinguished domains depicted in figure 1.

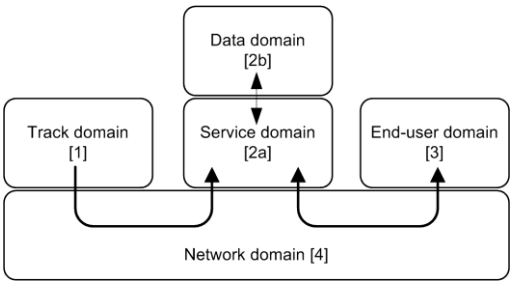

Figure 1 System domains

1) *Track domain*; this deals with the location of the runners and has the following requirements:

R1: The runners' positions are 'continuously' tracked (in 'real-time').

R2: The system uses exact (globally available) positioning techniques to get the runners position.

R3: The amount of resources in the tracking domain is limited.

2) *Service domain*; this is the central part of the system and is connected to the track domain as well as the end-user domain. It has the following requirements:

R4: There is a variety of end-user services (see also *R9*).

R5: The services offered to the end-users are location-based.

R6: The services are offered in 'real-time' to the end-users (providing up-to-date information).

3) *Data domain*; this deals with position data and the GIS part of the system. It has to fulfill the following:

R7: It attaches relevant geo-referenced information to the position data.

R8: It provides 'real-time' information for push and pull services.

4) *End-user domain*; this deals with the end-users using the system services. It has the following requirements:

R9: Services are based on different end-user's point of view.

R10: Various end-user networks and interfaces (user devices) are supported.

R11: Even if an end-user is only able to give rough position information, the offered services are location-based and spatially accurate.

5) *Network domain*; this deals with the transport of data between various domains and has the following requirements:

R12: A generally available network service is used, so no set-up time is required.

R13: The services are globally accessible.

6) *System internal* requirements:

R14: The system is built from existing off-the-shelf techniques.

R15: The system has a modular structure, in order to be easily extensible and scalable.

In this paper a 'real-time' service is defined as: *a service with a delay of at most a fixed amount of time*. The reason why we allow some positive (rather than zero) delay is that all systems and infrastructures being used suffer from delays. In order that the services are still attractive, we impose an upper bound (still to be determined) on the delay.

We now discuss various aspects, taking several design decisions along the way.

3.1 Tracking

To track runners, small (miniature) tracking devices are used, which should not hamper the runner's performance. However, these miniature devices have limited available resources. On the other hand, system requirements dictate that the system is to be built from existing techniques and is to be easily extensible. To meet these requirements following design decision is made:

Design decision D1: is to use the GSM network and the GPS system for tracking. These systems are widely available and many relatively cheap and small devices are on the market.

In order to track runners, some information needs to be transmitted from the tracking device to a central system. To meet this requirement the following design decision is made:

Design decision D2: The transmitted data must contain information to determine the objects identity, position and its time (for information synchronization purposes). In order to predict future data, course (direction), and speed information might be added. Except for the identity information all the necessary information can be collected from a simple GPS receiver. As unique ID for the tracking box, one can use the world wide unique IMEI from the GSM device.

Since the information needs to be provided in real-time, it is important to reduce network-delays while sending the data from the tracking device to the central application. Therefore, following decision is made:

Design decision D3: To send the data over the network an *unreliable protocol* (e.g. UDP) is the most suitable one. Although in this case a small amount of information will be lost during transmission and some data will be delayed, a large amount of data will arrive with minimal delay. Another advantage of this type of protocol is that there is less network data overhead and reduced (power) resources at the sending entity (because no connection needs to be kept alive). The reason for not using a reliable protocol is that these kinds of protocols can suffer from great network delays (especially in mobile environments) and therefore, will invalidate requirement R6.

Since the amount of information that needs to be sent is small (about 50 byte), not much bandwidth is required. Therefore it is best to use a package switching network (like GPRS, UMTS). These networks do not require maintenance of the connection, which save resources at the tracking device.

The number of position updates per time-unit determines how accurate and real-time the data is. To determine the amount of updates one needs to carefully take into account available network capacity, number of tracking devices (and other network devices) in the network and the battery capacity. To make the system attractive for the end-users, the maximal delay must be at most 15 seconds, so the update interval must be at least 15 seconds.

As a rule of thumb, to make the tracking device as simple as possible, it should be a thin client and leave the (fat) server to do the resource consuming work.

3.2 End-users

Because the described system is designed for a running race, four different end-user groups can be identified:

- Runners
- Organisation committee
- Spectators along the route
- Spectators located elsewhere

Not only these user groups are interested in different types of information because they have their own point of view on the race, but also they have different geographical positions.
Design decision D4: Since not all users are able to give exact location information, the system must be able to determine their position from other user input than position coordinates, for instance using street names and other visible reference points. To do so, following design decision is made:

Because the end users are mobile they access the services via different (mobile) devices that require different data representations. One can imagine that a spectator along the route is interested in which runners is coming next and that a runner participating in the race wants to know how much time he is in front of the next runner (pull service), while a traffic regulator is interested in when the first runner is approaching, to clear the traffic from the road at the right time (push service).

So the services offered by the system must take into account the end-users (geographical) location and their role in the race. The end-user must be able to select different type of services and how and when he wants to receive the service.

Design decision D5: The *service domain* is the unit responsible for identifying the end-user output format and transforming the service information into the right device format (so the end user is not bothered with it).

Some end user services that can be offered by the system are listed below:

Proximity services

- "Which runners will pass by this part of the route in the near future?"
- "Notify me when runner X is approaching"

Locator services

- "Where is running team X right now?"
- "What position in the race has team X right now?"
- "Notify me when team X passed spilt point 1"

As one can see, both push and pull services can be offered.

3.3 Providing real-time data

After data has been collected, it should be transmitted, received and stored into a database that is able to store large amounts of data and

quickly executes queries. To be able to provide real-time 'push' and 'pull'-services, the information needs to be sent to the end-user as soon as the data arrives and matches his requested service. To do so, the database should be able to store large amounts of data and execute the queries almost at the same time as new data is being inserted.

Design decision D6: Since we want to retrieve as much position data about runners as possible, the DB must have the ability to continuously execute the queries on streaming data [4]. Erwin [3] defines two different types of continuous queries, which are useful for this application; *time-based queries* for pull-based services and *changed-based queries* for push services.

For 'real-time' services only the most recent data is important. To determine this data the concept of a time-window is used resulting in the following design decision:

Design decision D7: Within the time-window lays the data, which is assumed to be recent. The size of this time-window depends on:

- speed of moving objects
- update frequency (from tracking devices)
- reliability of the data source (tracking devices), network and transmission protocol
- application-server load

The time-window handles the unsynchronised autonomous tracking devices, tracking data losses and delays. On the other hand it makes sure the latest information is shown to the end-users and out-dated data is removed (e.g., dropped out runner). Figure 2 illustrates the time-window concept.

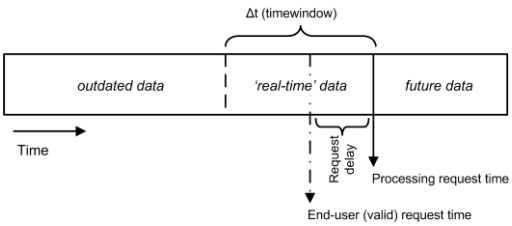

Figure 2 Schematic depiction of time-window

The time-window can be implemented by a (time-parameterised) query or a caching mechanism, which contains the 'real-time' data.

3.4 Visualisation

The next step is offering the provided services to the end-user, in a 'real-time' manner. Since the system will be accessed by various end-user devices, using different network services (which have their own limitations), the service needs to be adapted according to the end-user interface and network capabilities that the user is using to access the service (*see* D5). This adaptation is needed in order to provide the service to the end-user as soon as possible, by minimizing transfer delay, client loading time, and overall client capabilities.

To place the position information into its context, the runners' position coordinates need to be combined with some geographical information. Within the service domain there must be a process, which attaches geo-information to the runners' position information before the information is sent to the end-user.

4 System architecture

This section describes a system architecture to meet the given requirements, based on the taken design decisions. We address only an abstract architecture, a more refined description of the design can be found elsewhere [2].

Figure 3 is an overview of the main components of an application server that implements the central part of the system. It is connected to a communication network, which is the bridge between application server and the end-users and tracking devices. The application server consists of three main components:

- *Position receiver*; handles the incoming position data from the tracking devices.
  - At the network interaction point 1 (iap 1), data is received from the tracking devices.
  - At (iap 3) rightly formatted information is inserted into the database.
- *User request handler*; handles end-user requests and provides the requested service.
  - At (iap 2) requests are received and responses are returned.
  - At (iap 5) data request are sent and results are returned.
- *Query handler*; provides the information for the requested services that are retrieved from location DB and external (GIS) data sources.
  - At (iap 4) queries are sent and information is retrieved from the location DB.

- At (iap 6) queries to external data sources are sent and results are received.

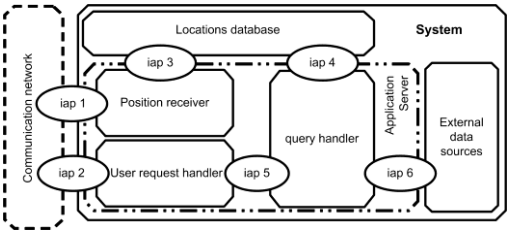

Figure 3 Overview of main building blocks and interaction points (iap) of the system

The building blocks are chosen in such a way that there is a clear separation of concerns. This has the advantage of realizing modularity. Since different building blocks communicate via the interaction points (iap), which are communication interfaces, a part of the system can be changed without affecting the other parts of the systems. A second advantage of this approach is that the application can be easily spread over multiple servers, which makes the application also scalable.

4.1 Position receiver

This part of the system is concerned with the reception of the data received from the tracking devices, and shields the other parts of the system from the protocols and data formats used by the tracking devices. It deals with data errors and different data formats and versions. It provides the information to the other components in one (predefined) uniform format.

4.2 User request handler

The user request handler deals with the interaction between the system and the end-user. It receives the service requests, retrieves the right information from the query handler and adapts the information in the appropriate output format for the end-user. In order to provide real-time LBS, this entity needs to determine following items about the end-users:

- device properties (to provide the information in the right format),
- current network capabilities, and
- current location.

To retrieve this information the system may use external data sources (e.g., a cell-id database), since not all end-user/user-devices are able to provide this information.

In order to handle many end-users and to keep service quality, we suggest using multiple user request handlers; this can be done because of the separation of concerns of the system building blocks.

4.3 Query handler

This part of the system is the access point to the information provided by the services that are offered to the end-user. It shields the other entities from the queries and protocols used by the various information sources. It also offers the information to the other parts of the system in a uniform format. This part is the main part of the system and provides the rough information needed to provide the various end-user services. This is done in two ways, on request from the user-request handler (pull service) or on a kind of subscription service (push service). When a condition of a subscription is met, the information is collected by the query handler and an event trigger is sent to the user-request handler, which provides this information in the right format to the end-user.

5 Conclusion

Based on some experiences with the tracking system BPS [1], we have given the requirements for a real-time location-based service system for running races. The requirements take the following into account:

- Different end user viewpoints need different end user (location-based) services (R4+R5+R7+R9).
- Different end-user use different devices and connecting networks (R10+R11).
- The system is to be built from existing off-the shelf building blocks (R12+R13), while taking into account the limited amount of available resources (R14+R3).
- The end-users are offered the (true) tracking information in ‘real-time’ (R1+R2+R6+R8).

These requirements are fulfilled by the following decisions:

- Provide LBS services (D4+D8) for different end-user groups using different devices (D5).
- Provide ‘real-time’ tracking information using exact positioning techniques with minimal delays (D1+D2+D3).
- Provide end-user ‘real-time’ pull and push services based on tracking information (D6+D7).

The system architecture is aimed at scalability and separation of concerns of the building blocks (R15). This allows easy

extending and updating of the system, optimizing various blocks for a specific task and spreading of the systems capacity. These properties will greatly affect the real-time-ness of the system.

References

[1] *D. Boekestein, S.P. Ekkebus, P.G. Uithol*, 'BPS – Bata Positioning System
A real-time online tracking system for the world's largest relay-race', LBS Symposium, TU Vienna, 2003
[2] *S.P. Ekkebus*, 'Real-time location based positioning system - a requirement study and design', Bachelor thesis, DB-group – University of Twente – The Netherlands, 2003
[3] *Christina Erwin*, 'Streaming data and continuous Queries', April 2002
[4] *D.B. Terry, D. Goldberg, D. Nichols, B.M. Oki*, 'Continuous Queries over Append-Only Databases', ACM Conference on the Management of Data (SIGMOD), 1992, page 321-330

Links

[L1] BPS system project website:
http://www.batalive.nl
http://eleanor.student.utwente.nl/bps
[L2] Champion chip:
http://www.champoinchip.com
[L3] Fitsense website:
http://www.fitsense.com
[L4] Competitio Inc.website:
http://www.competitio.com