ABSTRACT

The ubiquity of sensor data calls for automatic processing to extract valuable information. Realtime Locating Systems (RTLS) provide many parallel position data streams for interacting objects, and event-based systems are the method of choice to analyze them.

We demonstrate a distributed event processing system for position stream data from a Realtime Locating System used for a soccer application. Our system can deal with the insufficient knowledge on object and system behavior, and thus the event data loads at runtime. To do so, it dynamically adapts to the variations in the observed environment: events are ordered with respect to their delays, event detectors are reconfigured and migrated between nodes at runtime, and the system is scalable as the number of trackable objects and sensors changes. We demonstrate the efficiency of our system architecture and provide tools to visualize data and to configure detection units at runtime.

Categories and Subject Descriptors
C.2.4 [Computer-Comm. Networks]: Distrib. Syst.—Distrib. Applications; D.1.3 [Programming Techniques]: Concurrent Programming—Distributed Programming

Keywords

1. INTRODUCTION

With wireless localization techniques we can track moving objects in almost arbitrary environments. Realtime Locating Systems (RTLS) can track many objects simultaneously and provide accurate positions at high data rates [1].

By automatically processing this position information, we ease analyses in many fields of applications. Consider the quantitative analysis of sports games like soccer [2]. To detect meaningful data about players, teams and games, and to display information like percentual ball possession on television, we currently require several human observers that count certain events. Further scenarios arise in training. Cones are placed at specific points on the field, time keeping is assured by light barriers, and velocities are either calculated as a result of previous measurements or are directly determined with lasers. However, such setups are both time-consuming and error-prone. Thus, qualitative results are not objectively correct and are highly doubtful.

A low latency analysis of position data streams provided by an RTLS aids in the implementation of all such scenarios. The aim is to automatically derive events such as passes, ball possessions, or shots from the position data streams. In training, the system implements virtual time gates and sends their results to, for instance, a tablet PC on the field where the trainer can immediately use this information to improve the training or to compare the performance to previous exercises or to other players.

Event-based processing of position data streams is the way to go. Since many interesting incidents, i.e. events, depend on common lower-level events, it is obvious to calculate these basic events only once, and to form an event hierarchy from them. Fig. 1 shows such an event hierarchy to detect a blocked shot on goal. By splitting a pattern into various sub-events, we can reuse those sub-events for the detection of other high-level events. Lower software complexity, no coupling between event implementations, no memory synchronization, implicit parallelism, and better maintenance are further benefits that facilitate the development of event-based systems.

2. SYSTEM ARCHITECTURE

Fig. 2 shows our distributed event-based system (EBS) consisting of several data distribution services (DDS), i.e., the units that take microwave signals received by antennas to calculate the transmitters’ positions, and several nodes in
a network that run the same event processing middleware. The middleware creates a reordering buffer per event detector (ED), wrapped with a speculation unit to reduce latency. The middleware deals with all types of delays such as processing and networking delays or detection delays and does not need to know the complex event pattern that is implemented in the ED. EDs can hence be implemented in a native programming language or in some description language. The detector does not need to know on which machine other detectors are running nor their runtime configurations. At startup the middleware has no knowledge about event delays but just notifies other middleware instances about event publications and subscriptions by advertisements [4]. The middleware is therefore generic and encapsulated.

On top of the middlewares there are EDs spread over the available computing nodes. These EDs then work on an event-based interaction level and implement a message-dependence as, for instance, the one depicted in Fig. 1. In total, our system is used to detect over 500 different event types out of around 80 different EDs, depending on the scenario of use. The complexity of EDs varies from low-level detection to highly sophisticated algorithms.

Since the system and object behavior and thus the event loads are unknown a-priori, a static allocation and configuration of the EBS is rarely an optimal solution. Hence, our EBS implements dynamic low-latency reordering of events per event detector [3], migrates event detectors for latency optimization or system overload avoidance [5], and implements speculation to exploit unused system resources for further latency reduction [6].

3. DEMONSTRATION

We present details of our distributed event-based system and our tools that are used to process and visualize sensor data and events. On a distributed system of laptops we run both a virtual replica of the positioning and database system that is used to collect and store sensor data in the main soccer stadium in Nuremberg, Germany, and the distributed event-based system used to analyze the streams and to detect the events. On a tablet PC we demonstrate the tools and GUIs that are used by the trainers and professionals for qualitative analysis, see Fig. 3.

Visitors can design their own exercises and statistic graphs of the previously collected position and event data. Moreover, since the position streams from the stadium can be replayed to the EBS, visitors can also design and configure training scenarios for a realtime analysis as if the data would be gathered in the stadium. We also show our implementation to tackle the DEBS 2013 Grand Challenge [2].

4. CONCLUSION

We sketched the architecture of a distributed event processing system that works with high-volume position sensor streams and that can deal with the challenges and requirements of the sports applications. Its unique features are the basic building blocks that are geared towards low-latency, self-configuration, and reconfiguration at runtime.

5. REFERENCES