Relying on Wireless Sensor Networks to Enhance the RC-Gaming Experience

Pablo Guerrero*, Mariano Cilia** and Alejandro Buchmann**
*GK Enabling Technologies for Electronic Commerce and
**Databases and Distributed Systems Group,
Dept. of Computer Science, TU Darmstadt, Germany
{pguerrer,cilia,buchmann}@dvs1.informatik.tu-darmstadt.de

***Abstract*— Maturity of wireless sensor networks will lead to a world full of smart objects, and remote controlled toys are no exception. Following the growing popularity of multiplayer computer games, we envision a novel application that enriches the gaming experience by taking the digital multiplayer interaction into the physical world of remote controlled toys. We propose the development of an infrastructure that relies on wireless sensor networks as the glue that makes it possible that remote controlled toys can be used in multiplayer games, and provide a roadmap for the development of such an infrastructure.**

I. Introduction

Wireless sensor networks (WSNs) technologies let us start thinking about how to realize the vision of a world full of smart objects. This vision has typically been motivated by scenarios such as those in the military field, habitat/environment monitoring, agriculture or surveillance. However, a more common and major aspect in our life is *entertainment*, either in the physical or digital world.

An entertaining activity in the physical world is, for instance, to play with remote controlled (RC) toys. Nowadays, many types are offered such as cars, airplanes, helicopters, even robots with guns. These differ in complexity, starting from small, ready to use toys for kids, through assemble-yourself more complex toys, to complete hobbies for demanding enthusiasts. Playing with RC toys is very exciting, however, the gaming mode has not changed so far. The way they are used today consists basically of finding an appropriate spot (race track, open space, lake, mud, etc.) and play until no energy (i.e., batteries, fuel, etc.) is left. This is typically characterized by short playing sessions that vary from a couple of minutes to a few hours.

In the digital world, in turn, multiplayer computer games (over a computer network) have gained increasing popularity among players since the release of Doom in 1993 [1]. Single player modes, where the player fought against computer enemies, evolved into multiplayer modes like Co-operative or Deathmatch, where each player controlled his character from another side of the network. New *gaming modes* were conceived afterwards, like Capture the Flag, Domination, Assault, Last Man Standing or Invasion, to name some. These gaming modes have definitively leveraged the playability of existing computer games.

Our idea is to enrich the RC players' experience by taking the digital multiplayer gaming interaction into the physical world of the RC toys. Currently, there are many RC toys ready to be used with such games, however, no middleware is available that allows this kind of interaction. With some expertise in WSNs and multiplayer games, we can provide a gaming framework that enables the deployment of a broad spectrum of team-based, goal-oriented gameplay to participants. In this paper we propose the construction of a WSN infrastructure that glues both worlds, making possible new multiplayer game experiences for RC toy enthusiasts.

II. Enhancing the Multiplayer Game Experience with Onslaught

To exemplify the functionality of the infrastructure, we present a simple gaming mode called *Onslaught* [7], taken from the popular game Unreal Tournament [2]. Participants are divided into two opposing teams, their goal is to capture and hold strategic points, called *Power Node*s, in order to destroy the enemy's *Power Core*. Power Nodes are linked to each other and to a Power Core, forming a predefined *virtual network* (see Figure 1). Participants' toys start from their team's Power Core (e.g., the Red or the Blue Power Core) and advance across the gameyard towards the opponents' Power Core by conquering virtually connected Power Nodes.

Each Power Node has a *strength* attribute, whose value is initially neutral, and represents the intensity with which it has been conquered. Players can conquer them by *staying* for some seconds within a certain range around them. A Power Node can also be later neutralized and converted to the other team's color by taking the same action. In contrast, the Power Cores can not be *healed* back. Finally, each of these nodes has a light that indicates the team's color it currently belongs to; the light's intensity reflects the strength value.

When, for instance, the Blue team has conquered a set of Power Nodes such that they form a path starting at their Blue Power Core and ending at the Red Power Core, the latter can be conquered. As long as this restriction is fulfilled, the Red Power Core's strength can be weakened. However, if a Red team member breaks the path connection by neutralizing one of the intermediate Power Nodes, then the Blue team must either reconquer it or use an alternative path.

Obviously, two or more RC toys could be trying to conquer or defend a Power Node or Power Core at the same time, so players should expect impacts on the toys, in order to put each other out of the node's range. The game finishes when a Power Core is conquered, i.e., when its strength becomes zero.

III. An Infrastructure for Multiplayer Real Life Games

In order to allow RC toys to support team-based, goal-oriented gameplay like the example shown in the previous

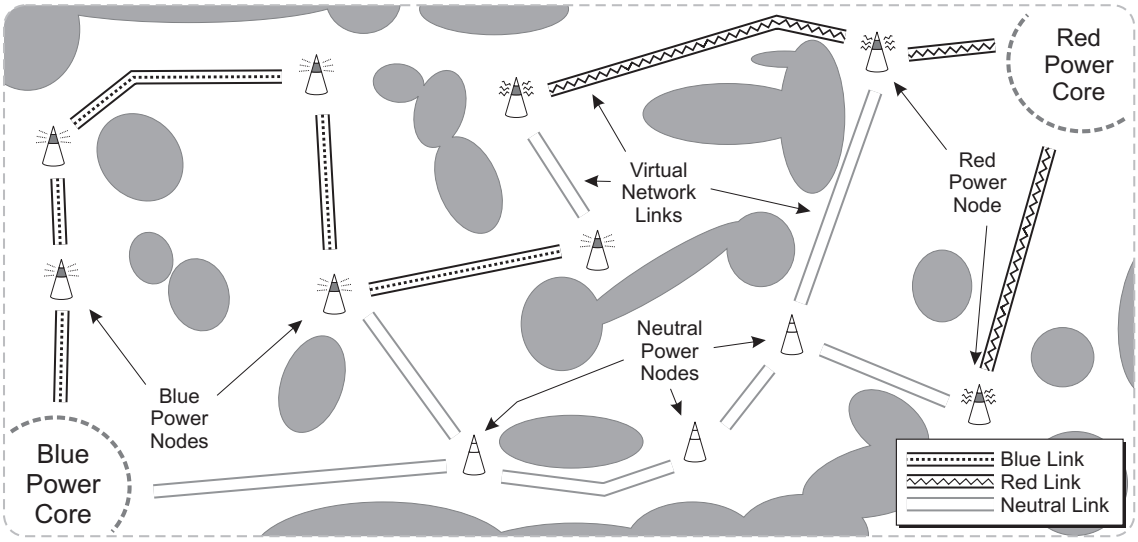

Fig. 1. A simple Onslaught map snapshot

section, an infrastructure is required. This work proposes the development of such an infrastructure, which binds three different domains, namely middleware for wireless sensor networks, digital multiplayer games and interactive toys. We have called this intersection *Multiplayer Real Life Games*, or MPRLG for short.

The set of rules that conform a game like Onslaught implies sensing and computing some data, as well as the ability to react on this sensory data. This is not only true for the RC toys, but also for the set of *game gadgets* deployed in the gameyard like the Power Cores and Power Nodes. *Game components* (i.e., both the RC toys and the game gadgets) enable the new gaming modes by signaling the different occurrences of the game and of course need some type of wireless connectivity. The collected data aids players in achieving the game goals. This is the point where the WSN infrastructure comes into play by providing:

1) a computational model to specify the game rules and assign them to the game components;
2) the means to disseminate the information between interested parties, like game components or controllers; and
3) a placeholder at the game components that triggers these rules and executes the associated actions.

In the next section we describe the architectural aspects we want to explore in the design of our infrastructure together with the open issues found, while in the subsequent section we focus on the communication aspects of the infrastructure.

IV. WSN Gaming Framework organization

The development of an infrastructure providing the features listed in the last section presents many technical challenges. The foundation of our design, which enables the game components to *sense*, *compute* and *communicate*, consists of *attaching* a wireless sensor node to each game component, in such a way this does not alter its physical mobility. For a prototype implementation, a sensor node must be identified that is appropriate both in capabilities and size. In this way, the proprietary, unidirectional communication channel used by the remote controllers to operate the toys remains unaltered. We proceed in this section describing other architectural aspects.

1) Gaming modes specification: As a framework for game developers, an important high-level issue faced in the infrastructure design is the selection of a convenient gaming mode *specification* language. We argue that the gaming mode's *logic* can be abstracted as a set of goals and restrictions. Therefore, a MPRLG can be modeled using an event-triggered rule language, in particular *Event-Condition-Action* (ECA) rules [8]. In this way, a gaming mode is modeled as a collection of rules that clearly specify when they should be triggered and what to do in reaction. At runtime, a *rule engine* component will trigger the corresponding rules and execute their associated actions. This approach exists in a platform-independent fashion and can be scaled down to small embedded systems [4]. However, no concrete implementation exists that runs on resource scarce devices like sensor nodes, and thus the reuse of existing rule engine systems is inhibited. The factors to consider include rule representation, storage and evaluation mechanisms. Furthermore, rule engines are basically stateless, hence game state must be maintained in a game-independent fashion. Hence, this state must be efficiently structured in order to optimize the (reduced) memory usage.

2) Infrastructure configuration and gaming mode deployment: Gaming modes normally require some initial setup and configuration, which occurs at two different levels. First, in a lower level and given the unreliable communication nature of WSNs, participants want to check the proper operation of the network before the game starts. One of the problems in this step is to have the nodes organize themselves into an operational network. This is usually handled by the networking protocols, for instance, by adjusting routing tables with local neighbors. Second, in the upper application layer, gaming mode dependent attributes must be set. The user interaction is required, for instance, to help provide unique identities (accessible by the application) to the game components, assigning them with different roles, or setting whatever fine-grained game parameters might need to be specified like overall game duration. The deployment phase can then begin, which could be done by distributing the corresponding role logic to each game component *over the air* using the wireless communication protocol, although this is known to consume

considerable energy from the node's power source.

3) Other game aspects: Many other aspects must be carefully engineered but because of space reasons will be discussed elsewhere. Between these we can cite the runtime interaction between the player and the game state (i.e., visualization interface); teammates interaction (for instance, send strategic commands to other teammates); usage of other game gadgets (e.g., like traps that penalize the toys by *blocking* them for some seconds); and finally usage of further sensors like accelerometers, energy (e.g., battery, nitro, gas, fuel) sensors, etc., that expand the gaming mode rules' power.

V. Game Components Communication

In order to update the game state, sensor nodes exchange messages in a many to many relationship. Due to the spontaneity of the game field settlement, an ad hoc, infrastructureless architecture is required. In our particular scenario, a gossiping-like traffic pattern is predicted: every game component (or *vantage point* as called in [12]) will generate unique information that needs to be communicated to the other components. This contrasts with the traditional sensor network scenarios where only one sink exists [3], [5].

1) Logical communication: at the top of the communication protocol, we would like to explore the suitability of a small footprint *publish/subscribe* (pub/sub) data dissemination mechanism. This would provide a useful indirection to distribute game events across the game components. Generated data might then be filtered, aggregated, and eventually disseminated towards interested consumers such as the controllers.

2) Network protocol: network functions must provide an efficient routing scheme that reliably conveys messages to and from other network nodes, across multiple short hops, and some considerations have already been taken by [6]. Moreover, game components change their location fast and constantly, making the overall topology highly dynamic, which requires a strong support for mobility. Some form of disconnected operation (e.g., buffering) could be involved as well in order to deliver messages generated while an RC toy was not in the reach of other game components.

3) MAC layer: traditional MAC protocols for WSNs typically trade off energy consumption for latency and fairness [10], [11]. While power efficiency is always an important consideration, the trade off is reasonable when sensors are under the control of a single (monitoring) application. However, our RC toys' sensor nodes virtually represent individual participants, with equal rights to produce or consume information in order to update the game state. Therefore specialized MAC protocols that consider node mobility should be inspected, such as [9].

4) Physical layer: currently, sensor networks are being architected using one of 802.15.x PHY layers, which dictate the theoretical data rates and energy consumption values. We expect to be able to decide for one of these by adjusting all the knobs with a top-down approach as described before.

VI. Conclusions

In this paper we have presented the idea of integrating the physical world of RC toys with the virtual world of multiplayer computer games, which we have called MPRLGs. This entertainment is made possible by relying on a WSN gaming infrastructure. The idea is to enhance the gaming experience by providing team-based, goal-oriented gameplay to the toys.

The set of requirements for these games was outlined by describing the Onslaught gaming mode as a concrete example. This game introduced the concept of game gadgets and their ability to detect toys in their surroundings. These requirements raised the necessity of having a pluggable WSN ad hoc infrastructure that allows to specify, configure and deploy rules at the game components, as well as to disseminate the game-related data that updates the game state.

Currently we have sketched an initial infrastructure design that adequately faces the aforementioned challenges, while raising others for each area of its layered design. In the uppermost logic level, we use ECA rules to describe a MPRLG application layer, together with a rule processor that reacts to specified game events. The game components communicate logically using a publish/subscribe system, which provides a convenient interface for the ECA rules to disseminate game related data.

Bearing some resemblance to how advances in computer games have pushed high-end computer graphics, we believe MPRLGs can also drive the evolution of WSN technologies. In particular, the project provides an interesting testbed to try, evaluate, stress and further refine ideas and actual algorithms and other different aspects related to wireless sensor networks, reactive systems and publish/subscribe notification services.

VII. Acknowledgments

We would like to thank Patric Kabus, Kai Sachs, Jan Steffan and Falk Fraikin for their collaboration, ideas and reviews of several drafts of this paper.

References

[1] Doom (c) by id Software, Inc. http://www.idsoftware.com.

[2] Unreal Tournament (c) 1999-2002 by Epic Games Inc. Created by Epic Games Inc. in collaboration with Digital Extremes. http://www.epicgames.com, http://www.digitalextremes.com.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: a Survey. *Computer Networks*, 38(4):393–422, 2002.

[4] J. Antollini. Implementing an Active Functionality Service on Different Platforms. Master's thesis, UNICEN, Argentina, July 2005.

[5] A. Boulis, C. Han, and M. B. Srivastava. Design and Implementation of a Framework for Efficient and Programmable Sensor Networks. In *Procs. 1st. Intl. Conf. on Mobile Systems, Apps. and Svcs.*, pages 187–200, New York, NY, USA, 2003. ACM Press.

[6] C. P. Hall, A. Carzaniga, J. Rose, and A. L. Wolf. A Content-Based Networking Protocol For Sensor Networks. Technical Report CU-CS-979-04, Dept. of Comp. Cs., University of Colorado, August 2004.

[7] Epic Games Inc. Unreal Tournament 2004 gaming modes. http://www.unrealtournament.com/ut2004/modes.php, 2001.

[8] N. W. Paton, editor. *Active Rules in Database Systems*. Springer, New York, 1999.

[9] H. Pham and S. Jha. An Adaptive Mobility-Aware MAC Protocol for Sensor Networks. In *Procs. IEEE Intl. Conf. on Mobile Ad Hoc and Sensor Systems*, pages 558–560, FL, USA, 2004. IEEE Inc.

[10] T. van Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Protocol for WSNs. In *Procs. 1st. Intl. Conf. on Embedded Networked Sensor Systems*, pages 171–180, NY, USA, 2003. ACM Press.

[11] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for WSNs. In *Procs. 21st. Intl. Annual Joint Conf. of the IEEE Computer and Communications Societies, INFOCOM 2002*, NY, USA, 2002.

[12] M. Zoumboulakis, G. Roussos, and A. Poulovassilis. Active Rules for Wireless Networks of Sensors & Actuators. In *Procs. 2nd Intl. Conf. on Embedded Networked Sensor Systems*, pages 263–264, New York, NY, USA, 2004. ACM Press.