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

Pablo Guerrero[1], Mariano Cilia[2] and Alejandro Buchmann[2]

[1] GK Enabling Technologies for Electronic Commerce, and
[2] Databases and Distributed Systems Group,
Dept. of Computer Science, TU Darmstadt, Germany
{guerrero, cilia, buchmann}@dvs1.informatik.tu-darmstadt.de

Abstract. The state of the art in pervasive computing technologies will slowly allow turning into a world full of smart objects, and remote controlled toys are no exception. Following the growing popularity of multi-player computer games, we envision a novel application that enriches the gaming experience by taking the digital multi-player 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 enables using remote controlled toys with multi-player games, and provide a road map for its development.

1 Introduction

Advances in pervasive computing technologies are slowly allowing us to improve different aspects of our daily activities. One of such aspects is *entertainment*, which can be done either in the *physical world* (i.e., sports, table games, etc.) or in the *digital world* (i.e., by means of computer games).

A very entertaining activity in the physical world is to play with remote controlled (RC) toys, and nowadays many types are offered such as cars, trucks, airplanes, boats and even robots with guns. These RC toys differ in complexity, starting from small, ready to use toys for kids, to assemble-yourself more complex toys, to complete hobbies for demanding enthusiasts. The toys are operated remotely by their owners with a *controller*. The wireless radio connection is *unidirectional*, usually split in multiple channels at several frequencies

Playing with RC toys is very exciting, however *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. battery, fuel, etc.) is left. Due to the toy's energy consumption, this is characterized by short playing sessions that vary from a couple of minutes to a few hours.

On the other hand, in the digital world, multi-player computer games (over a computer network) have gained increasing popularity among players since the release of Doom in 1993©. Single-player gaming modes, where the player fought against computer enemies, evolved into multi-player modes like Deathmatch or Co-operative. Computer opponents could be replaced with human player's characters, controlled from another side of the network. Later on, new gaming modes

were conceived like Last Man Standing or Invasion. 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 multi-player 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 infrastructure is available that allows this kind of interaction. In this paper we propose the usage of wireless sensor networks (WSNs) as the gluing infrastructure that makes possible that RC toys can be played as a multi-player game.

2 Enhancing the Game Experience with Onslaught

In order to exemplify the infrastructure functionality, we show a concrete gaming mode called *Onslaught* [10], inspired by the popular game Unreal Tournament©. Within Onslaught, participants are divided into two opposing teams. The gameyard contains two *Power Cores* that act as base stations (e.g., the Red or the Blue Power Core). These, in turn, are linked to each other through several strategic points, called *Power Nodes*, forming a predefined *virtual network*. The team goal is to conquer the enemy's Power Core, while defending theirs'.

The game rules are simple. A Power Node can be conquered when a toy *stays* for some seconds within a certain range around it. Moreover, a player can conquer a Power Node if and only if there is a path composed by other conquered Power Nodes connecting it with its team's Power Core. Power Nodes have a *strength* attribute representing the intensity with which they have been conquered, and indicated with a color light. An opponent can also neutralize and convert a Power Node to its team's color by taking the same action. Power Cores can be similarly conquered, however they can not be *healed* back.

Players' toys strategically advance across the gameyard towards the opponents' Power Core by conquering virtually connected Power Nodes. 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 simultaneously try to conquer or defend a Power Node or Power Core, 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.

Figure 1 shows a simple Onslaught map snapshot, as well as the described entities. By applying our knowledge in WSNs and multi-player games, we can offer new gaming modes to the participants. Our aim is to provide a gaming framework that enables deploying a broad spectrum of team-based, goal-oriented gameplay. Next, we describe the application requirements and provide a road map for its development. We finalize by providing a summary of related work and the conclusions.

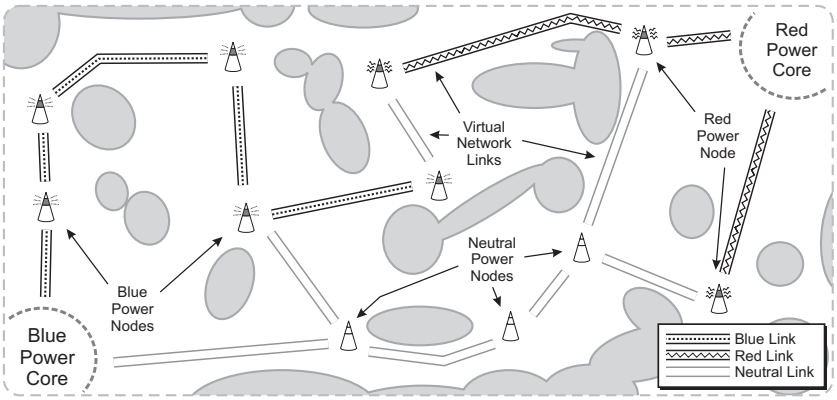

Fig. 1. A Simple Onslaught Map

3 Multi-player Real Life Games' Requirements

In order to allow RC toys to support gaming modes like the example shown in the previous section, an infrastructure is required. This work proposes the development of such an infrastructure, which binds three different domains, namely pervasive computing, multi-player games and toys. We have called this intersection *Multi-player Real Life Games*, or MPRLG for short.

The small set of rules for a game like Onslaught requires sensing and computing some data, as well as the ability to act on the sensory data. This is not only true for the RC toys, but also for the set of *game gadgets* like the Power Cores and Power Nodes deployed in the gameyard. *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 (e.g., *Red Car #7 conquered Blue Power Node #3*) and of course need some type of wireless connectivity. This is where the 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 the interested parties, like game components or controllers; and
3. a placeholder at the game components that triggers these rules and executes the associated actions.

The first challenge is to enable the game components to sense, compute and communicate. We propose to *attach* a wireless sensor node to each game component and controller. In this context, a node consists of a processor, some memory, a wireless radio and a power source. This scheme presents the advantage of keeping unchanged the proprietary unidirectional channel between controllers

and toys, used by players to manipulate them. The wireless sensor nodes use another radio frequency to convey game-related data that coexists with the remote controlling channel. We proceed in this section describing important non-functional attributes and constraints that challenge the infrastructure design.

3.1 Gaming Modes Acquisition, Configuration and Deployment

The gaming mode's *logic* can be expressed as a set of rules, which could be contained by a *Game Repository*, accessible through the web. A *Game Configurator* such as a PDA can be used to download a gaming mode. Normally some initial setup and configuration is required, which occurs at two different levels. First, given the unreliable communication nature of the WSN, participants want to check the proper operation of the network before the game starts. Nodes must organize themselves into an operational network, for instance, adjusting routing tables with local neighbors. Second, 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, assign them with different roles, or setting fine-grained game parameters (e.g. overall game duration, maximum strength, etc.).

After players agree to a gaming mode, deployment of the corresponding role logic into the toys' and gadgets' nodes occurs, using the wireless protocol, although this is known to consume considerable energy from the node's power source. Participants could join the game dynamically, as long as the gaming mode rules are deployed into their toys' node first.

3.2 Game-Player Runtime Interaction

The infrastructure should allow participants to visualize the game state. For instance in the Onslaught game, participants could visualize the virtual network of Power Nodes conquered so far and their strength. Ideally for the players' mobility would be to embed a small size screen into the remote controllers, or have regular PDAs. This allows each player to monitor the information of his *personal* toy, such as the location or speed, and would give players within a team the opportunity to be physically distant. Through this interface, participants can also send strategic commands to other teammates, for instance, by indicating a meeting point in his screen's map with a pen. The command would be sent through the nodes mesh.

3.3 Beyond Onslaught

In general, information generated by further sensors like acceleration, temperature, light, orientation, wind and energy (e.g., battery, nitro, gas, fuel) sensors, signal strength indications, localization and context-aware algorithms, etc., could feed the game rules. In addition, toys could provide an I/O interface where the wireless node can be plugged to query its intrinsic hardware integrity. In this

way, part malfunctions, engine temperature, or any other internal readings can also enrich the game rules.

The game logic, defined by means of rules, should clearly define which information can be shared with others and what must be made private (i.e., visible only by the player or its team), particularly since the transport mechanism could convey the events through any wireless node in the mesh (i.e., a team-mate or an opponent's sensor node).

4 Infrastructure Design and Challenges

The infrastructure required to support the described gaming modes must face many technical challenges. In this section we show the considerations that lead to our layered infrastructure design. These layers are illustrated in Figure 2.

Application Layer. Several key abstractions were identified in the top layer. The *Game Manager* provides an interface to handle deployment details as mentioned in Section 3.1. This component contains an *id* that uniquely identifies the node it resides in from the rest. The id can either be obtained from the operating system or an algorithm can be used. It also stores the node's *role*, which can be given by the properties of the node itself (e.g., has particular sensors).

This layer also deals with the gaming mode logic. We argue that a MPRLG can be modeled using an event-triggered rule language such as Event-Condition-Action (ECA) rules. These rules clearly specify when they should be triggered and what to do in reaction. For instance, in the Onslaught game we may have:

```
Event:     ON [ CarDetected(car) ]
Condition: IF [ ∃ neighbor Power Node pn /
               {CurrentTeamColor(pn) = TeamColor(car)} ]
Action:    DO [ AdjustStrength(TeamColor(car)); PublishState(); ]
```

This rule exemplifies the easiness with which the Power Node's logic can be described using ECA rules. The Event clause indicates that the rule must be triggered when the node detects a '*car*' in the proximity of the Power Node. The Condition clause states that the action must be executed only if there is a neighbor Power Node '*pn*' that is currently conquered by '*car*'s team color. Finally, the Action clause contains an list of activities, in this case the adjustment of the Power Node's strength attribute, and the dissemination of its state. Note that although this rule is triggered by an event sensed locally by the node, it could also by triggered by an event received from a neighbor node.

During runtime, a *rule engine* triggers the rules as a result of incoming events, checks the required conditions, and executes the associated actions. This architecture is based on an *Active Functionality Service* [4], which is platform independent [2]. Roughly, the ECA Manager component administrates the ECA-rule registration and activation, and relies on other elementary services when rules are triggered, like the Event, Condition and Action Services. This approach is

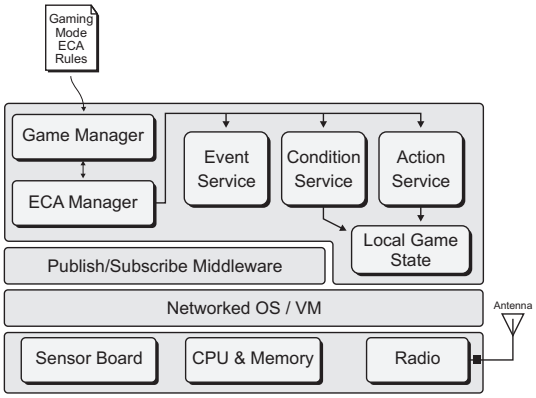

Fig. 2. Infrastructure Architecture

scalable to small embedded systems. However, although it has been proposed for WSNs [13], we are not aware of concrete implementations.

Finally, the rule engine is basically stateless, hence we provide a Local Game State component, which stores information required by the application services. Therefore, game ECA-rules must be adequately arranged such that only a minimum set of game state is kept in the sensor nodes.

Game components' logical communication. In order to update the game state, wireless nodes communicate with their peers, exchanging game-related events in a many to many relationship. The asynchronous communication between nodes can benefit from using the publish/subscribe (pub/sub) paradigm, since it naturally decouples producers and consumers, makes them anonymous to each other, and allows a dynamic number of publishers and subscribers. Generated data might be filtered, aggregated, and eventually disseminated towards interested consumers, like peers or controllers. Due to the spontaneity of the gaming field settlement, an ad hoc, infrastructureless communication is needed. Using an access point-like infrastructure would not only increase the deployment effort, but also require more power at the nodes to transmit data over longer distances. Instead, nodes route data to interested consumers across multiple short hops. In addition, game components are in constant movement. This implies that a) available neighbors at one time will likely change, and b) messages might be generated while a game component was not in reach of others. As a result, the nodes conform a mobile ad hoc network that must include some form of disconnected operation (e.g., buffering).

Foundation Layer. A foundation layer (e.g., OS, VM or simply embedded libraries) is required that provides an abstract interface to the underlying hardware, timers, task scheduling and so on. Also the medium access control (MAC)

protocol is normally included in this layer. However, since WSNs MAC protocols typically trade off energy consumption for latency and fairness [5], therefore specialized protocols that consider mobility should be inspected, such as [11].

Physical Layer. An appropriate hardware platform must be identified for this infrastructure. Sensor hardware was already discussed in Section 3.3. We foresee that the first wave of WSN platforms (e.g., those with 8 bit processors) will not suffice our required processing capabilities. This is already being addressed, e.g., by [7, 12]. Finally, current WSNs' designs use one of the 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.

5 Related Work

Interesting work has been done by investigators trying to bridge digital and physical games with pervasive technologies. In [1], an outdoor multi-player game called *Unmasking Mister X* is presented. Wearable computers and sensors are used to determine who is 'Mister X', based only on readings from his wearable sensors. In *Can You See Me Now?* [8], some players run around a real city's streets (tracked by GPS) while others move an avatar in an online 3D representation of the same city. Physical players attempt to 'catch' online players by chasing their (virtual) location. *Treasure* [6] is a game that exploits the lack of connectivity in wireless networks. Players move outside wireless coverage to collect virtual 'coins' and then move back into an area with high network strength to 'upload' the treasure to a game server. However, these games run on equipment with relatively large resources (for instance PDA's with GPS). *Trove* [9], in contrast, implements a multi-player game on a WSN, where participants try to reach a hidden treasure. Each player is assigned with a mote which transmits game packets to a base station. Players loose lives whenever their sensor readings values exceed a configurable threshold. A game supervisor exists which performs identification, sensor calibration and coordination of the game in a similar fashion to our Game Configurator. Another related project, although not meant for gaming, is *CotsBots* [3]. It integrates off-the-shelf motes with a commercially available mini RC car, whose RC functionality and other original electronics are extracted and replaced by an autonomic behavior board that controls the car's engine, providing an experimentation platform for distributed robot systems.

6 Conclusions

In this paper we have presented the idea of integrating the physical world of RC toys with the virtual world of multi-player computer games, which we have called MPRLGs. The gaming experience can be enhanced by providing goal-based, team-oriented gameplay to the participants. This entertainment is made possible by relying on a WSN gaming framework. Its requirements were sketched

with an exemplary gaming mode, which introduced the concept of game gadgets and their ability to detect toys in their surroundings. This raised a pluggable WSN ad hoc approach that allows the specification, deployment and execution of rules at the game components, as well as the dissemination of game-related data. And later we have discussed an architecture facing its technical challenges.

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 and measure different ideas and algorithms related to wireless sensor networks, reactive systems and publish/subscribe notification services.

7 Acknowledgments

We would like to thank to Patric Kabus, Kai Sachs and Jan Steffan for their collaboration and draft reviews. A particular thank to the members of the ESF Scientific Programme MiNEMA, where first steps of this research were discussed.

References

1. S. Antifakos and B. Schiele. Bridging the Gap Between Virtual and Physical Games using Wearable Sensors. In *Procs. of ISWC'02*, Seattle, USA, Oct. 2002.
2. J. Antollini. Implementing an Active Functionality Service on Different Platforms. Master's thesis, Faculty of Sciences, UNICEN, Tandil, Argentina, July 2005.
3. S. Bergbreiter and K. Pister. CotsBots: An Off-The-Shelf Platform for Distributed Robotics. In *Procs. of IROS'03*, Las Vegas, USA, October 2003.
4. M. Cilia. *An Active Functionality Service for Open Distributed Heterogeneous Environments*. Ph.D. Thesis, TU Darmstadt, Germany, August 2002.
5. T. Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Procs. of SenSys*, pages 171–180. ACM Press, 2003.
6. M. Chalmers et al. Gaming on the Edge: Using Seams in Pervasive Games. *2nd. Intl. Workshop on Gaming Applications at Pervasive 2005*, May 2005.
7. R. Adler et al. Intel Mote 2: An Advanced Platform for Demanding Sensor Network Applications. In *Procs. of SenSys*, pages 298–298. ACM Press, 2005.
8. S. Benford et al. Can You See Me Now? *ACM CHI Transactions*, 2005.
9. S. Mount et al. Trove: a Physical Game Running on an Ad-Hoc Wireless Sensor Network. In *Procs. of sOc-EUSAI*, pages 235–240, Grenoble, France, Oct. 2005.
10. id Software Inc. UT 2004. `www.unrealtournament.com/ut2004/modes.php`, 2001.
11. H. Pham and S. Jha. An Adaptive Mobility-Aware MAC Protocol for Sensor Networks. In *Procs. IEEE Intl. Conf. MASS'04*, pages 558–560, FL, USA, 2004.
12. R. Smith, C. Cifuentes, and D. Simon. Enabling Java for Small Wireless Devices with Squawk and SpotWorld. In *Building Soft. for Pervasive Computing '05*, 2005.
13. M. Zoumboulakis, G. Roussos, and A. Poulovassilis. Active Rules for Wireless Networks of Sensors & Actuators. In *Procs. of SenSys*, pages 263–264, NY, USA, 2004. ACM Press.