

# Parrotfish: Task Distribution in a Low Cost Autonomous ad hoc Sensor Network through Dynamic Runtime Reconfiguration

Dionissios Efstathiou, Konstantinos Kazakos, Apostolos Dollas<sup>†</sup>

*Department of Electronic and Computer Engineering  
Technical University of Crete  
Chania, 73100, Greece  
Contact e-mail: dollas@mhl.tuc.gr*

## 1. Introduction

Reconfigurable resources can provide substantial field programmable capability in the wireless autonomous nodes of an ad-hoc sensor network. The intersection between distributed systems and reconfigurable ones is a topic that has not yet been studied methodically. In previous work [1] we have presented some problems and challenges in distributed reconfigurable systems and our on-going research in these topics. The problems are compounded by the need for low cost solutions and support for multi-vendor systems. Thus, there is a need to reconfigure or partially reconfigure individual nodes in order to alter system behavior, or circumvent non-fatal errors.

The Parrotfish project is a low cost, distributed environment for (partial) reconfiguration of distributed field programmable systems, e.g. sensor networks. In this paper we present architectures and results in which the wireless nodes of a distributed system can undergo runtime task-reversals under triggering from external conditions, using Bluetooth as a low cost wireless medium. The project gets its name from the small fish found in Florida waters, which can change gender as needed under group dynamics.

## 2. Wireless communication issues

The choice of Bluetooth as the wireless medium vs. other options (IrDA, WiFi) was not obvious [2], and was determined after consideration of several parameters.

**Flexibility:** In a wide-scale, computationally demanding environment the exclusive usage of vendor specific FPGAs may decrease the complexity and increase the compatibility of the system but it may not offset its reduced efficiency. Thus, the existence of a heterogeneous (multi vendor) multi-FPGA system may sometimes be necessary and therefore a vendor-independent protocol needs to be used for the wireless communication, with vendor-specific configuration support (e.g. file transfers) at the node boundaries. The communication protocol must be able to sustain

exchange of processed data and configuration files in an efficient, reliable and fault tolerant manner.

**Addressing:** In a point-to-point system, the addressing protocol can be considered as a trivial issue. On the contrary, in a point-to-multipoint network topology, the following issues need consideration:

- At any given time, one node is by default the master and the rest are slaves.
- The master must have the ability of multicasting and broadcasting information to the entire network.
- Since the network is decentralized, each node must be able to become the master node of the network.
- The addressing protocol must provide an identification mechanism for all nodes in the network.
- The addressing protocol fully depends on the number of nodes deployed in the network.

## 3. Parrotfish node architecture

The Parrotfish node is partitioned into three inner-communication layers, initially proposed in [1].

• The “lower” layer (layer1) is the physical and data-link layer. It consists of the Bluetooth module and its controller. This layer secures the efficient data transport between the nodes and is referred to as data link layer. It fully supports multicasting and broadcasting

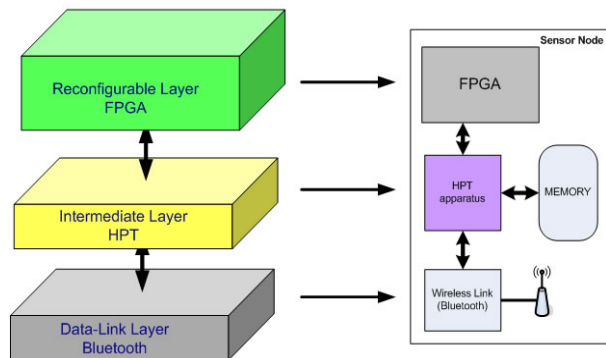
- The “middle” layer (layer2) is the control medium of the sensor node.
- And the “upper” layer (layer3) is the reconfigurable section.

Layer2 acts as a centralized source for run-time (re)configuration of the reconfigurable processing-section and an intermediate between the reconfigurable layer and the rest of the network. It consists of the HPT (Hardware Programmer & Tester) [3], an 8-bit microcontroller and a memory storage unit.

All incoming data from the network that reach the node are collected by layer1. In turn, the middle layer “parses” the received data and according to their use, they are forwarded to the reconfigurable section (layer3) or kept as needed. Vice-versa, layer2 collects processed

<sup>†</sup>also at ITRI, Wright State University, Dayton, OH, USA

data and requests from layer3 and either redirects them to the upper layer for transmission to the network or processes them. It also has the ability to dynamically (re)program or read back the configuration bit stream of the node's reconfigurable section (layer3).



**Fig 1.** Layers of transparency

The architecture of the Parrotfish node offers a level of transparency between the different layers. Each layer only communicates with its adjacent layer.

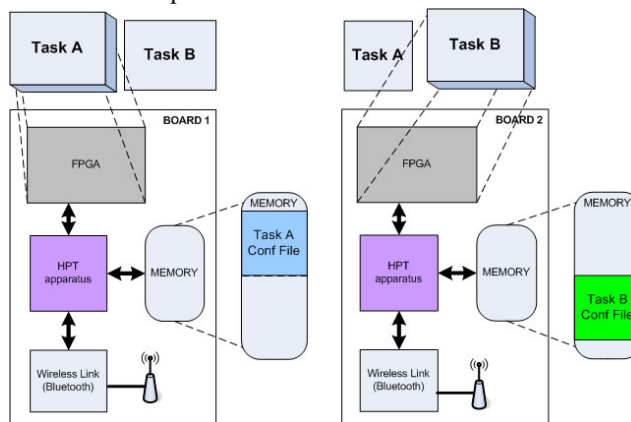
#### 4. Case Study

The purpose of this case study is to demonstrate the Parrotfish project's capabilities mainly in the flexibility that can provide in a distributed sensor network. The present Parrotfish project is comprised of two identical nodes. Each node is capable of performing a single task, "Task A" or "Task B", as depicted in Figure 2. The first task is the implementation of the "game of life" whereas the second task simply drives a 7 segment display panel. Each task is completely independent of the other and its execution requires one full configuration of the reconfigurable section (layer3) through the HPT with a configuration file stored in the system's memory module.

The protocol in this network dictates that each node must, at any time, execute different tasks. If a node due to an external factor must change task, the other node must act accordingly. Since the nodes only contain the configuration bitstream of the currently executed task, the HPT apparatus is responsible for "swapping" the bitstreams between the two nodes. Another rule in this protocol is that the "task swapping" mechanism must be dynamic. In other words, each node must be reconfigured with the other node's latest execution instance. Task execution must continue from the exact point where the other stopped and vice versa.

In order to achieve dynamic reconfigurable computing it is necessary to allow tasks to be preempted. On preemption the state of the task should be saved either by readback (if the FPGAs support it) or through some other method. Significant work on this area is mentioned in [4]. The "task swapping" procedure is

controlled by the "middle" layer. The HPTs in the nodes take a "snapshot" of the FPGAs' state, through readback, and store them into their memories respectively. When the first node verifies the error-free transmission of Task B's configuration file, it starts transmitting Task A's configuration file respectively. As a following stage, the HPT apparatus in each node reconfigures the FPGA with the newly arrived task. As a result, the two nodes have completely switched tasks, with each node continuing from the exact point where the other has left.



**Fig 2.** Parrotfish nodes.

#### 5. Acknowledgements

We would like to thank Professor Nikolaos Bourbakis, Director of the ITRI, for hosting Prof. Dollas during his sabbatical, Teleca Comtec, Inc. and Ericsson, SA, for substantial donations of Bluetooth development systems, and Dr. Brian von Herzen for his useful suggestions, including the project name Parrotfish.

#### 6. References

- [1] A.Dollas, D.Efstathiou, G.Vernardos, E. Polytarchos, K.Kazakos, "On Distributed Reconfigurable Systems: Open Problems and Some Initial Solutions", Proceedings of the Annual IEEE Symposium on Field Programmable Custom Computing Machines (FCCM), April 17 - 20, 2005, Napa, California, USA, (poster).
- [2] J.Haartsen, M.Naghshineh, J.Inouye, O.Joeressen, and W.Allen. "Bluetooth: Vision, goals, and architecture", ACM Mobile Computing and Communications Review, 2(4):38-45, October 1998.
- [3] A.Dollas, D.Efstathiou, T.Kyriakides, "A Universal Low Cost Run-Time and Programming Environment for Reconfigurable Computing.", IEEE International Workshop on Rapid System Prototyping, 2003, pp. 2-8.
- [4] A.Ahmadinia, C.Bobda, D.Koch, M.Majer, J.Teich, "Task scheduling for heterogeneous reconfigurable computers", Integrated Circuits and Systems Design, 2004. SBCCI 2004. 17th Symposium on, 7-11 Sept. 2004, pp.22 - 27.