Towards the Better Design, Implementation, and Simulation of Embedded Wireless Systems

Sally K. Wahba
School of Computing, Clemson University
sallyw@cs.clemson.edu

1 Problem Statement

Embedded network systems are used to support a variety of applications, from social networking [1] to saving lives in the battlefield [2]. Where size does not correspond to simplicity, these systems, composed of tiny devices, impose several challenges in the design, implementation, and testing phases of their life cycle. This section describes the challenges we propose to solve – one in each stage.

**Design Challenges.** The first challenge is ensuring reliable performance measured by radio link quality, defined here as the ratio of the number of packets received to the number of packets sent. This is not an easy task, mainly due to the low-power and low-cost radios used in these systems. Network links experience significant variation in link quality, leading to unpredictable and undesirable performance characteristics. There is inadequate support for predicting link quality in the early stages of system design and implementation, resulting in systems built on ill-understood foundations.

**Implementation Challenges.** The second challenge is developing flexible and reusable software. Design patterns have become an integral part in desktop system development, but they are rarely used in embedded wireless system development. The main hindrance to the applicability of existing desktop patterns is that embedded wireless systems are fundamentally different from desktop systems. They are interrupt-driven and are subject to severe resource constraints, precluding the application of most desktop patterns. We believe there are many patterns for embedded wireless system development yet to be identified.

**Testing Challenges.** The third challenge is testing embedded wireless systems that use self-stabilizing algorithms. Self-stabilizing algorithms are widely used in embedded wireless systems given that transient faults are a common occurrence in such systems. The problem is that field-testing of applications requires significant financial resources and human effort. Simulation provides a more feasible means of testing. Unfortunately, there are no simulators for simulating the types of self-stabilizing algorithms used by embedded wireless systems. Further, available simulators are unable to capture the physical environment in which systems will be deployed, which may lead to post-deployment operation problems. A simulator is needed to identify any faults at an earlier stage.

2 Preliminary Plan

We provide a preliminary plan to solve the challenges in Section 1. **Input from the ICSR community is mostly required for the implementation and testing challenges.**

**Design Challenges.** We plan to develop the first empirical model that accurately predicts the link quality of embedded wireless systems. We will consider
different factors that impact link quality: radio transmission power, inter-node distance, physical obstructions, interference, and the deployment site topography. The model will be based on experimental results from a range of environments. Regression analysis will be used to develop the model from the data collected from these experiments. Questions that need to be answered include: How long should an experiment be run to gather data for each experimental setting? Will the model be applicable beyond the hardware used in the experiments?

**Implementation Challenges.** We plan to identify design patterns in embedded wireless systems by studying expert-crafted systems. We will study major design challenges in embedded applications and how available systems address them. When we encounter a recurring problem, a candidate pattern will be identified. Once identified, we will study different systems to check how the designers solved this problem. If the solutions share core design characteristics, the candidate pattern will be codified. Questions that need to be answered include: How will the identified patterns be evaluated for correctness and flexibility? Will we restrict our study to systems written in certain programming languages?

**Testing Challenges.** We plan to develop the first simulation framework for self-stabilizing algorithms. To accurately simulate these algorithms, we will incorporate the link quality model discussed above to account for the network deployment environment. Incorporating the model will be an option for the developer using the framework so that the framework will be suitable for distributed systems in general and embedded wireless systems in particular. The framework will also provide a library of sample algorithms. Questions that need to be answered include: What is the tradeoff between the framework accuracy and efficiency? How will the link quality model be incorporated in the design?

### 3 Work Completed

We discuss the work completed in each of the proposed components.

**Design Challenges.** A preliminary piece-wise model of link quality has been developed. The model predicts link quality as a function of transmission distance and radio power [3]. We found that a single empirical model to predict link quality was not feasible as link quality generally falls into 3 regions: low-, mid-, and high-quality. The model includes linear, polynomial, and exponential formulas resulting from linear and non-linear regression analyses.

**Implementation Challenges.** We have identified two patterns applicable to embedded wireless systems. The first, Basestation Dictionary, is used to overcome the space, bandwidth, and computation capacity limitations of embedded devices when representing large constants, such as strings. This is achieved by mapping these large constants to smaller ones that act as keys used by the embedded application. When the basestation receives the keys from an embedded device, it checks its map to decipher the keys and appropriately use their corresponding (larger) constants. The second pattern, Control-Flow Mediator, aims at simplifying the control flow and synchronization logic of applications by linearizing interrupt-triggered events and minimizing execution in interrupt context. This is achieved by setting the appropriate program state in interrupt context, without actually handling the event. A mediator function is introduced
to periodically select the event to be handled based on the state variables. The mediator runs periodically in an infinite loop or a time-triggered fashion. Applying this pattern enables the system to miss fewer interrupts since the application does not spend as much time running in interrupt context. Additionally, since the program control-flow is centralized in one location (the mediator function), it is easier to understand and analyze the application logic.

**Testing Challenges.** We have developed a general purpose simulation framework for self-stabilizing algorithms. The simulator backbone, sample applications, and a graphical user interface have been implemented. The simulator currently supports self-stabilizing algorithms, but does not take into account the link quality model. The framework supports algorithms commonly used in embedded network systems, such as anonymous leader election in a tree and breadth-first spanning tree. The framework includes different network graph generators, such as random and alpha-beta graphs. The framework design uses different design patterns, such as the Observer and Factory Method patterns.

### 4 Evaluation

This section discusses how each of the proposed contributions will be validated and evaluated.

**Design Challenges.** The datasets and link quality model will be validated for accuracy. The experimental datasets will be validated through repeating sample experiments in the same environment and obtaining consistent datasets. The link quality model will be evaluated by running the experiments in different environments and checking how the model adapts to such variations.

**Implementation Challenges.** The identified design patterns will be validated for applicability. We will study more expert-crafted systems to find applications of the patterns. We will also develop controlled experiments to test each of the identified patterns as we divide a group of novice programmers into two teams to solve the same problem. Only one team will apply the pattern under test. After solving the problem, the application software will be assessed to determine the flexibility and reusability of the solution. The team that used the pattern will be surveyed and asked to evaluate the usefulness of the pattern.

**Testing Challenges.** The simulation framework will be assessed for accuracy and scalability. Accuracy will be validated and evaluated by simulating a network and then physically deploying it to compare the results of the actual performance to the simulated performance. Scalability will be evaluated by simulating different applications using different environments and network sizes.

### References