Concurrent Embedded Power Control

Matthew Bajzek and Dr. Dennis Brylow
Marquette University

Introduction

Marquette University's Systems Laboratory requires a robust automated power solution capable of ensuring that its routers and other equipment are always in the desired on/off power state. These states are set based on requests sent via users on the laboratory's computers or automated scripts.

The laboratory currently utilizes a module with both hardware (a remote power controller) and software which dictates its behavior.

Prior to this project, the system did not include any means of controlling the flow of user calls to the remote power controller, nor did it address the system's many potential errors.. Figure 1 shows the interactions between the system's hardware and software prior to this project's changes.

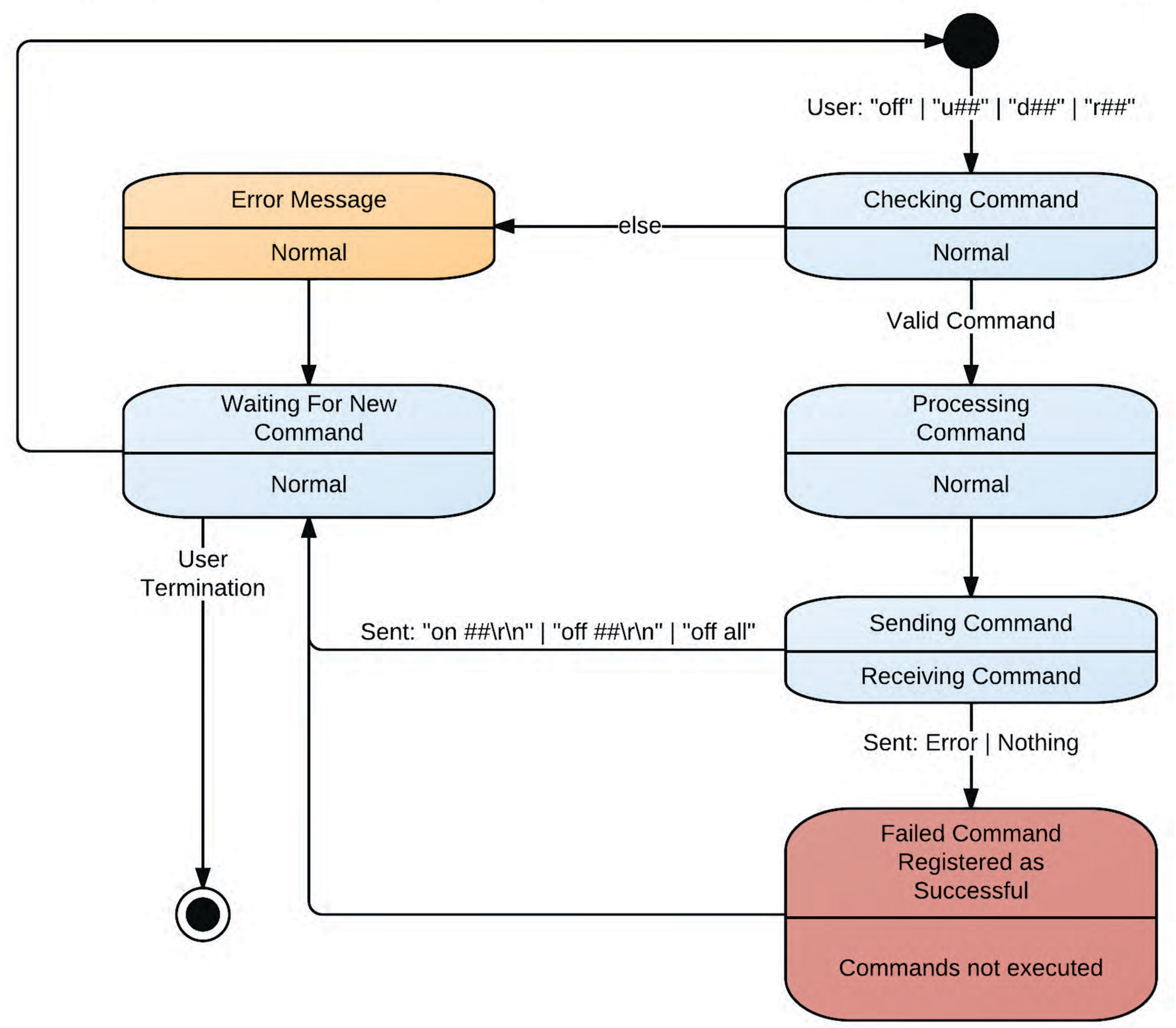

Figure 1: State Diagram For Previous System Flow

Objectives

Given the potential frequency and concurrency of user requests (many calls can be made near-simultaneously from different users), the system must be capable of regulating the communication flow between its hardware and software to ensure that all commands are allowed to execute before new requests are forced upon the hardware. As Figure 1 demonstrates, the previous system failed to diagnose or correct any unexpected behavior it encountered. This project's goal was to refine the system to self-diagnose and self-correct any problems it encountered during execution.

Methods

After determining that the primary source of rebooting error was timing (the system failed to regulate commands sent within 0.015 seconds of one another and completely ignored these errors), we began the process of searching for all conceivable places in the code that could produce errors. This search produced the state machines shown in Diagrams 1 and 2, which were refined to prevent any hard errors, infinite loops, or dropped calls.

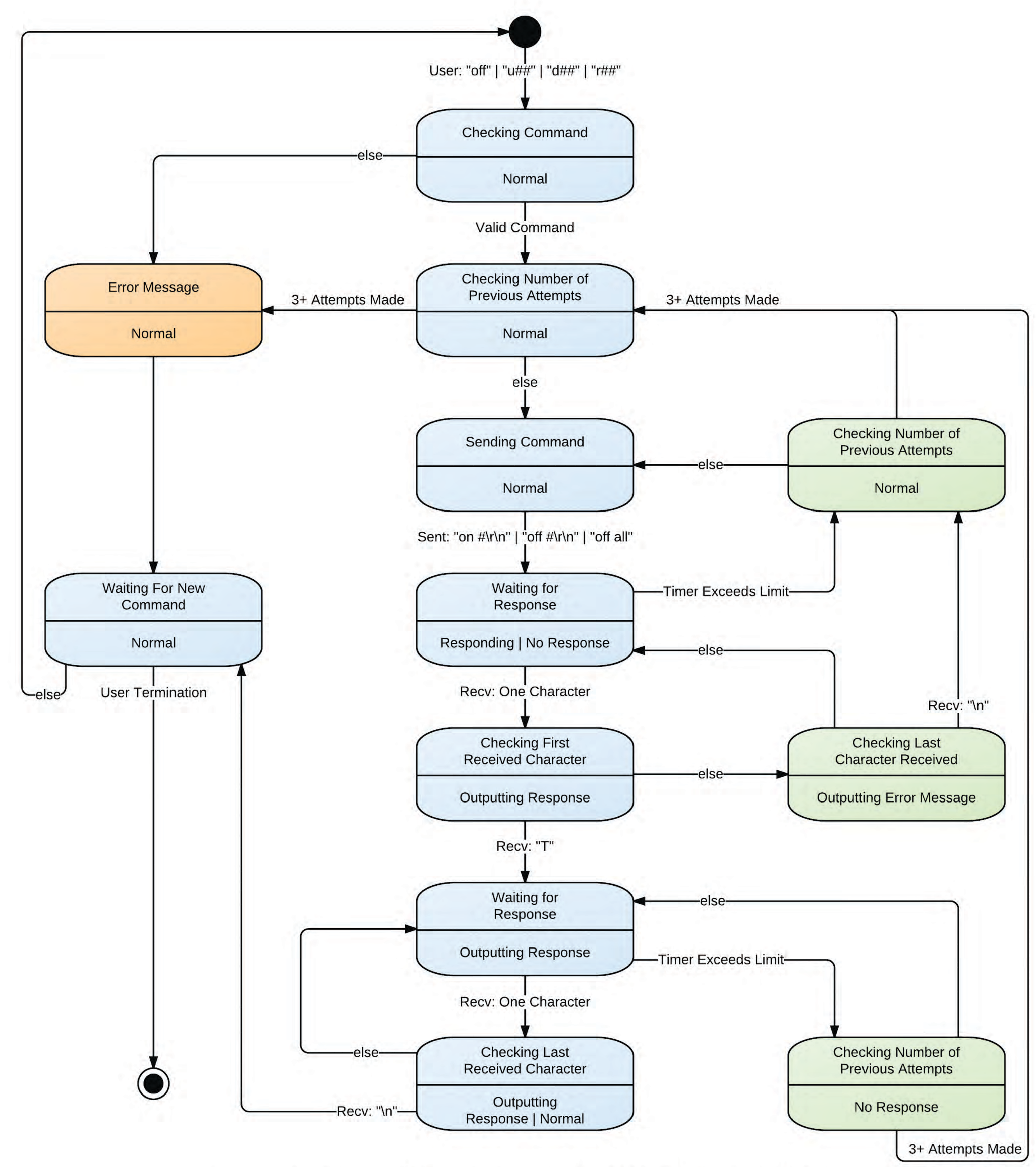

Figure 2: State Diagram For Updated System Flow

The overall design approach we employed while making the new state machine was to craft a series of loops to prevent the system from moving on to any new tasks before receiving hard confirmation (generally in the form of specifically-formatted response messages) that previous tasks had either succeeded or failed.

By waiting for previous requests to succeed before sending new ones, the updated software prevents the overlap that plagued the previous version. This dynamically allocates delays as necessary, eliminating the need for the user to wait in between requests.

Results

In limited testing which will be expounded upon for our paper, the updated system proved capable of receiving and executing a large number of commands made from multiple users simultaneously. Testing suites with up to fifteen simultaneous users (which consistently fail on the previous system) run without accident, and we have been unable to isolate a dropped call on the latest version.

Figure 3 shows the number of calls processed during a one-second period in which forty-five commands were sent (one after another) by a single user to a rebooter board separated from other hardware. This simplified test allowed for clearer comparison than a concurrent setup with routers and other hardware active, as the previous code's accuracy radically deteriorated in such a setting (to the point of being graphically unpresentable).

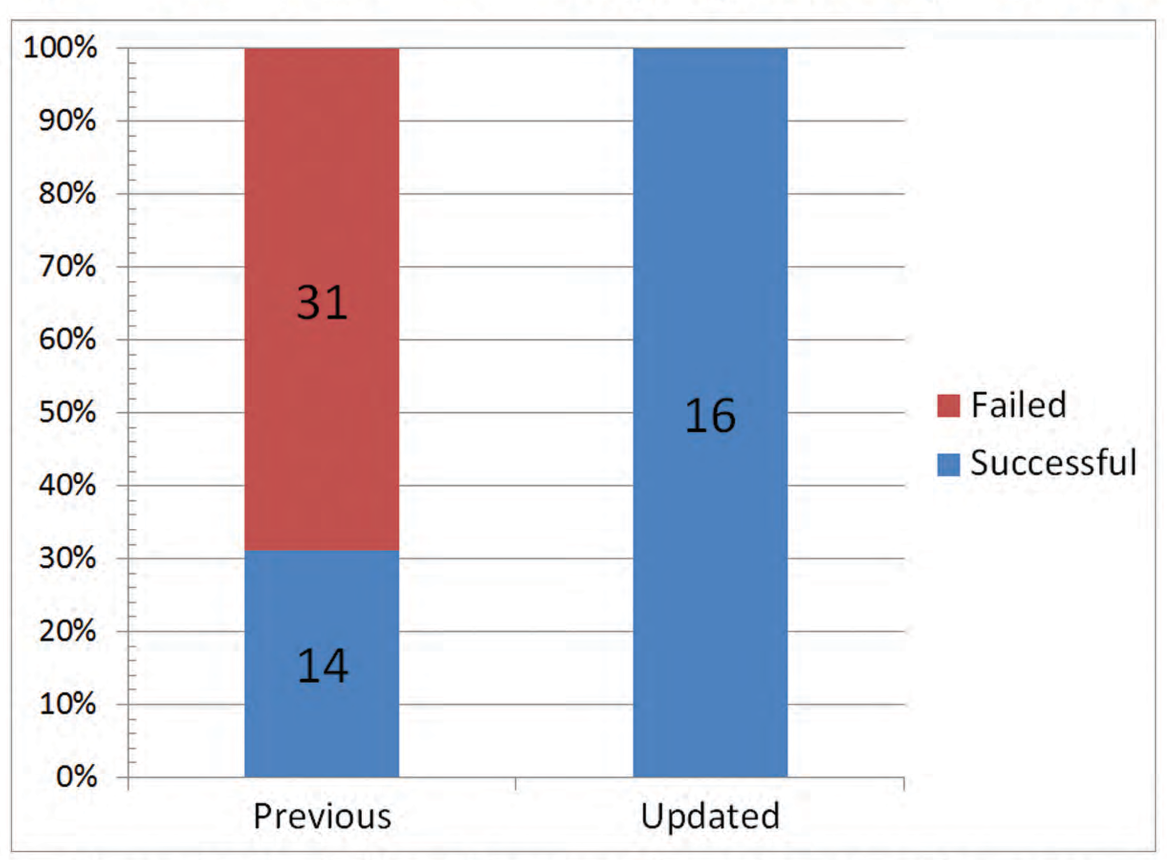

Figure 3: Single-User Request Throughput and Accuracy

Conclusion

The updated power system has proven itself far more efficient and accurate than its predecessor thus far, and will continue to do so as refinements are made and its use becomes more intensive.

Future Work

The updated system has been tested successfully under a variety of circumstances, but the true test of its efficiency will occur when it is utilized by truly large groups of users simultaneously during the academic year. This will provide useful data regarding the system's throughput that will allow further refinements to be made.

A truly dynamic queue could be added to the code, allowing the system to bypass routers' delay requirements by sorting commands it receives and separating reset requests into on/off commands. This would radically improve efficiency by reducing delays.