DataMule Design Report Summary

Group 24: Sam Praneis, Michael Oltman, Haider Basharat Ali, Sonya McKean

1 Overview

DataMule reimagines wildlife tracking by turning animals, drones, and patrol vehicles into a living, mobile network. Instead of paying thousands for limited satellite connections, our delay-tolerant system lets collars “store, carry, and forward” gigabytes of rich ecological data whenever contact is possible. The result: cheaper, longer-lasting, and far more detailed insights into animal behavior, delivered even in the most remote of places on Earth. With DataMule, we move beyond patchy signals toward an “Internet of Wildlife” that helps with data collection on wildlife while expanding the boundaries of computer networking.

2 The Design Stage

This third step of compiling the structure of the software that makes up DataMule mainly comprised of identifying Classes through the various means learned in class. While brainstorming got us a baseline (as seen in the Initial System Analysis section), the particulars were lacking. The authors believe that the Boundary Condition Analysis was especially useful, due to the fact that DataMule software will be run on systems which are in a particularly hostile environment. Damage is expected, so the software ought to be prepared. Once classes were identified and further designed, the rest of the paper went by quicker. Every section that was relevant to DataMule were filled out in a way the authors believe to be satisfactory. The glossary and index sections encapsulate anything one could wish to look up. Admittedly, the authors might have over-indexed as certain terms appear very often in this report, causing more than a dozen results to appear.

The one section that is less than satisfactory (at least to one author) is the bibliography. The bibliography is somewhat improperly formatted. Past a certain point, they are organized in a much different way, broken up into sections and unnumbered.

3 The Final Report

3.a The First Stage of the Report

The first section of this report is dedicated to the early planning of DataMule as software. A brief overview of the initial idea that caused DataMule to be chosen is given: the idea that a delay-tolerant networking system used in animal tracking collars might save researchers in the field money and offer reliability that best-effort networking simply doesn’t. Armed with the idea, we analyzed the state of this field as it stands. Satellite connections are reliable, but slow and expensive. Manual retrieval is consistent but labor intensive and potentially dangerous. LoRaWAN or similar networks have good bandwidth and lower cost than Satellite, but their reach doesn’t reach everywhere, especially everywhere wild animals are.

With a clear opportunity to improve upon what exists, we continued the very early plannin process. We defined our scope, identified possible and likely stakeholders as well as their relative priorities, defined our terms to be used for the rest of the project, noted all relevant mandatory constraints, and finally, listed the assumptions we were bringing into this project. With that, we were ready to advance to the second stage.

3.b The Second Stage of the Report

With the project ready to get further underway, we set our sights on gathering requirements. We first designed some core, mandatory use cases. In particular, these were: the gathering of a Low-Tier Node's data by a Mid or High-Tier Node, the transmission a Mid-tier node's telemetry back to the home High-Tier Node, the viewing of data through a frontend UI by a researcher, the handling of inevitable data transfer failures due to spotty connections or otherwise not detecting a higher-tier node, the distribution of Time and Keys to all Low and Mid-Tier Nodes to ensure every device is synced and kept secure, and updating our software.

These use cases ultimately guided the creation of our requirements. Once they were chosen, the description of requirements under each of the provided categories were easy to ideate. Inparticular, ideating requirements for their physical operation in an unoptimal environment generated plenty of non-functional but extremely useful requirements. Once requirements were generated, we simply needed to define acceptance tests to ensure each of them were satisfied. While this was a lengthy process (so much so that the acceptance tests are shuffled off to the end of the paper in their own section), the process was nonetheless critical, allowing us to finally begin ideating the design of our software.

3.c The Third Stage of the Report

With use-cases, requirements, and acceptance tests in hand, diving head-first into design couldn't quite begin. Our design needed to meet the requirements we'd set out, but we also needed to identify design goals to optimize for, otherwise we'd just push our first idea out.

The design goals we focused on were: efficiency (monetary and operational), dependability, security, and iterability. The rationale for each is fairly simple. Monetary and operational efficiency is the entire reason we believe this project has legs on the market. Dependability was another major factor in this project being marketable, as LoRaWAN and other networking structures do not have the capability to maintain this dependability in the wilderness where the animals would be. Security is a natural thing to optimize for, as if this design is insecure, we put animals, users, stakeholders, and our reputation in danger. Finally, iterability, or the ability for our software structure to evolve and be maintained, is critical going forward. If the software falls out of compatibility with the latest, greatest platforms, we'd lose our niche, and the needs of our end users may change with time (especially given the Sword of Damocles that is climate change).

With our design goals set, we analyzed pre-existing software solutions much closer than we had in stage 1. We identified many of the weaknesses we'd expected, and used their structure to build up a mental map of what sorts of classes we'd need. This early, initial design was simple, lacking any defined methods or variables, but still showed us where to start. From there, we modeled more use cases, and used them to show what sorts of classes would be needed, and how they were interact. This further iterated our design and gave us plenty of new components to add. With components in hand and a better idea of the structure, we divvied up DataMule into 3 major subsystems: Field Data Acquisition, Network Ingestion, and a Centralized Backend for User Access.

Naturally, you may see a few things missing from these systems and their proposed classes. Inparticular, there's no subsystem nor any mention of the frontend UI, nor does this account for any of the many boundary conditions that our software would be put under. Once the greater system was designed and the classes defined, we begun wrapping up. We defined potential risks, costs, replacements, current issues, and ideas implementation. Finally, a retrospective, a glossary, and hey, all those acceptance tests we shoved off until now.