RoboTractor

Jeremy Wright
Arizona State University
jlwrigh1@asu.edu

Arun Balaji Buduru
Arizona State University
abuduru@asu.edu

David Lucero
Arizona State University
dwlucero@asu.edu

***Abstract*—The need for the automation of vehicles is increasing due to improved efficiency and reduced operational costs. Even though the law requires human presence in self-driven agricultural vehicles, recent trends show that as our confidence in these self-driven vehicles grow, these requirements on human presence will be waived. Once we remove humans from control of these vehicles, secure signaling and control takes center stage as there will be no human to troubleshoot if the signal and control systems are compromised, making vehicles vulnerable to be stolen, damaged, or otherwise harmed. The scope of this project is to build a secure signaling and control system with mechanisms to drastically reduce if not eliminate these attack vectors. The main tasks in this project are to build a front-end interface for users to give instructions built upon a Representational State Transfer (REST) architecture, set up Extensible Messaging and Presence Protocol (XMPP) server and client with required authentication and data integrity check mechanisms, and build a simulated automated vehicle, capable of interfacing with the XMPP components and front-end.**

***Index Terms*—Secure signaling and control, remote management, agricultural vehicles automation, self-driven vehicles**

I. Introduction

Our goal in this project is to build a secure signaling and control system to enable the effective management of the agricultural vehicles. The primary problem to be addressed in this project is to securely interface and/or communicate between user and agricultural vehicles using XMPP and REST protocols. This is very important because authentication of vehicles and data integrity, here the path plan and location feedback transmission needs to be secured and protected from eavesdroppers. This will prevent attackers from compromising and/or stealing these vehicles. In this project we use Django to handle the front end interface for the users. We use python libraries to interface between user and path generator server where we use XMPP and REST protocols, and XMPP clients to control the agricultural vehicles. We expect to have a comprehensive tool that translates the abstract user inputs into executable actions for agricultural vehicles in a secure manner.

II. System Models

A. System Model

RoboTractor will leverage the Django Web framework [1] , to realize the required interfaces. Figure 1 describes the connection of these components. The combination of XMPP and REST in RoboTractor is a demonstration of how to extend the existing HTTP development environment i.e. ”the web of things” into a stateful protocol. HTTP is by design stateless. XMPP on the other hand is a stateful streaming connection between two clients. In this case the Tractor and a command server. To achieve this mesh we will leverage the Bidirectional-streams Over Synchronous HTTP standard protocol [2]. BOSH provides a standard mechanism to operate the streaming XMPP protocol efficiently over an HTTP connection. This is essential for a scalable webservice.

B. Software

1) XMPP Server: RoboTractor uses the ejabberd [3] XMPP server as it provides an existing Python interface, to integrate with the rest of our Python based ecosystem.

2) REST Interface: TastyPie provides REST [4] by extending the existing Django Models.

3) BOSH Interchange: Punjab is an Django plugin implementation of BOSH [5]. In addition to BOSH, this library combines the ejabberd Users with Django Users to provide a single authentication and authorization framework. While this demo project will have a single user type, this combination

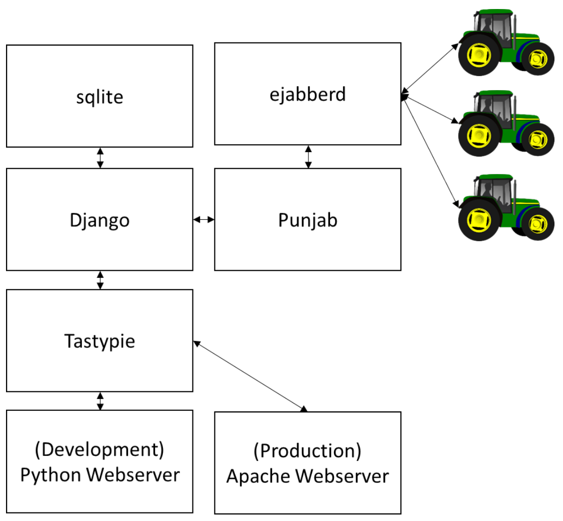

Fig. 1. Software Component Block Diagram

is critical to maintain proper use management and least-privilege authorization.

III. Project Description

Implementation is divided into 2 primary phases (Marked as milestones on the attached gantt chart). The first phase is an integration phase connecting the various libraries and off the shelf components. It is critical that this integration does not weaken, but rather strengthens the security properties provided by each independent component.

The second phase is the extension of these components to provide command and control of agricultural vehicles. For schedule mitigation, David and Arun are scheduled to begin this phase before the framework is assembled. Once the framework is in place, David and Arun's interface designs may be merged with the overall system.

A. Task 1 : Development Environment

The development environment consists of configuring all the tools needed to portably work with the software package. This will involve installing project dependencies, and deployment scripts to allow all members of the team to work effectively together. The complete environment will be stored in git.

B. Task 2: XMPP Implementation

XMPP Implementation is a configuration task to setup the ejabberd server and connect it into the HTTP Framework. Once this is in place the XMPP Clients may start working.

C. Task 3: Demo Tractor

The Demo tractor is the complementary component to the Frontend UI. This is the virtual machine, a piece of software which simulates a real tractor, or agricultural vehicle.

D. Task 4: Frontend UI

The "single-page" web application is the modern design methodology to web apps today. Leveraging this design architecture the Frontend will query the REST API to draw the position of all tractors within a Google Map context. This task has 2 high level sub tasks:

1) Path generation
2) Map rendering

Path generation is the primary deliverable of this project. The user shall be able to input a path for a given tractor to drive. The Tractor will then receive this path over the REST to XMPP gateway. The UI may then periodically query the position of the tractor and update it on the Google Map.

E. Task 5: Server Backend

The Server Backend is the critical component which plumbs all the components together. Extensive knowledge of Django will make this task easier. As it will require linking multiple components together in a orchestrated fashion.

F. Task 6: REST Interface

The REST interface will server the primary means of interacting with the site. The Web interface will exist as a "single-page" app who leverages AJAX principles over this REST API.

G. Project Task Allocation

Jeremy Wright will serve as the project lead since he has past experience with Python based web applications.

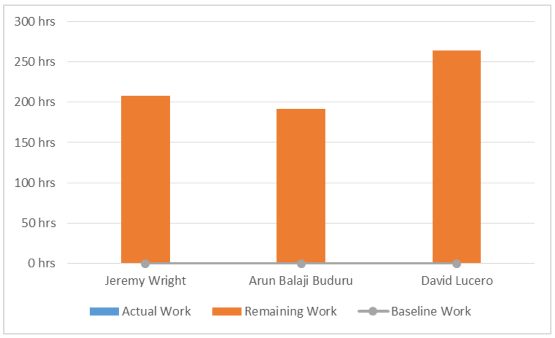

Fig. 2. Resource Allocation

H. Deliverables

1) Web application for sending control commands to remote machine, and displaying their current location
2) A Simulated vehicle capable of acting on XMPP command send via the Web interface.
3) An administration panel for configuring vehicles.
4) API documentation for using the REST interface as an external service.
5) API documentation for using the XMPP interface to act as a vehicle.

I. Project Timeline

The attached project timeline (generated from Microsoft Project) describes the overall tasks of the project.

IV. Risk Management of the Project

Several potential issues have been identified that may pose risk to successful completion of this project. These risks have been identified in Table I. Along with the description, ratings have been assigned to each risk identified, and potential mitigation strategies are described.

V. Conclusion

In this proposal we intend to build a remote control system leveraging existing internet technologies XMPP and REST. Future work based on this approach may include asset management for corporate farm management.

VI. Acknowledgment

We would like to acknowledge the ASU VLAB for their support of tools and services for demonstration and development of this project.

References

[1] (). Django, GitHub, [Online]. Available: https://github.com/django (visited on 02/06/2014).

[2] I. Paterson, D. Smith, P. Saint-Andre, and J. Moffitt. (Jul. 2, 2010). Bidirectional-streams over synchronous HTTP (BOSH). This specification defines a transport protocol that emulates the semantics of a long-lived, bidirectional TCP connection between two entities (such as a client and a server) by efficiently using multiple synchronous HTTP request/response pairs without requiring the use of frequent polling or chunked responses., [Online]. Available: http : / / xmpp . org / extensions / xep - 0124 . html (visited on 02/06/2014).

[3] (). Ejabberd community site | the erlang Jabber/XMPP daemon, [Online]. Available: http://www.ejabberd.im/ (visited on 02/06/2014).

[4] (). Toastdriven/django-tastypie, GitHub, [Online]. Available: https : / / github . com / toastdriven / django - tastypie (visited on 02/06/2014).

[5] (). Twonds/punjab, GitHub, [Online]. Available: https://github.com/twonds/punjab (visited on 02/06/2014).

TABLE I
RISK MANAGEMENT

Risk Description	**Risk of Failure**	**Consequence of Failure**	**Mitigation Strategy**
Connections between components must be secure	Low	Product would still function, but would lack security information assurance will suffer	Plan to include security on component connections ahead of time. Follow best practices (SSL/TLS)
Evaluation of Project depends on proper vehicle agents	Medium	System is unable to be tested if vehicle agents are inoperable or incorrect	Vehicle agents will be designed first to ensure compatibility with all other components. Thorough review and testing of component to ensure proper functionality
Improper configuration of components	Medium	Components do not communicate properly with each other. System functionality is void	Thorough reviews and testing of components to ensure proper functionality and communication
Incorrect PKI implementation and/or configuration	Medium	Puts all identified security domains at risk. PKI used becomes useless. System becomes vulnerable to outside attacks i.e. MITM	Follow best practices and standards. No re-invention of the wheel
Service Uptime (web reliably)	High	Product is unusable if network connections are not operable	Redundancy. Cloud hosting to alleviate hardware reliance