Volttron: An Agent Platform for the Smart Grid
(Demo)

Jereme Haack, Bora Akyol, Brandon Carpenter, Cody Tews, Lance Foglesong
Pacific Northwest National Laboratory
Richland, WA, USA 99352
{jereme, bora}@pnnl.gov

ABSTRACT
VOLLTRON™ platform enables the deployment of intelligent sensors and controllers in the smart grid and provides a stable, secure and flexible framework that expands the sensing and control capabilities. VOLTTRON™ platform provides services fulfilling the essential requirements of resource management and security for agent operation in the power grid. The facilities provided by the platform allow agent developers to focus on the implementation of their agent system and not on the necessary "plumbing" code. For example, a simple collaborative demand response application was written in less than 200 lines of Python.

Categories and Subject Descriptors
I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms
Design, Experimentation

Keywords
Agent platform, Smartgrid

1. INTRODUCTION
The future power grid will have many distributed assets including distributed generation, responsive loads and automation at the distribution system. The distributed nature of the future power grid provides a natural fit for the application of agent based systems. As sensors and controls spread throughout the smart grid, new techniques and technologies must be brought to bear to make effective use of them. VOLTTRON™ is a distributed agent platform that is being developed at Pacific Northwest National Laboratory of Richland, WA, USA as part of the Future Power Grid Initiative. We discussed VOLTTRON™ features and implementation in great detail in [1] and [2] and will summarize them in Section 2. In this paper, we will discuss the demonstration testbed we developed to show the capabilities of the platform in a realistic environment. The VOLTTRON™ testbed is based on commercial off the shelf equipment and (so far) emulates plug-in electric vehicles and electric water heaters. The details of the testbed will be covered in Section 3. We will conclude by discussing the lessons we learned while building and using the testbed.

2. VOLTTRON™ Platform
This section summarizes the architecture and design details of the VOLLTRON agent execution platform as discussed in [2]. The platform is illustrated in Figure 1. VOLTTRON™ exists between the operating system and the agent execution environments (AEE). As shown in the figure, VOLTTRON™ supports multiple AEEs such as Java, Python, and platform-specific binary objects.

The VOLTTRON platform consists of communications services (CS); resource manager (RM); authentication and authorization (AA); directory services (DS); agent instantiation and packaging (AIP); and information exchange bus (IEB) modules. The AA module includes a policy and trust store as well as an optional policy manager function. The IEB module includes a local store (LS) to provide non-volatile storage for agents. AIP is responsible for packaging, instantiation, and coordination of agents' movement. AA module provides validation of agent payloads, authenticates peer platforms, and handles public and private credentials. When an agent payload comes in from another platform, it is handled by the Agent Instantiation and Packaging (AIP) module. AIP sends the authentication information to the AA module. The AIP sends the execution contract to the Resource Manager that checks that the platform has the required resources. If both checks pass, the AIP then passes the payload to the Agent Execution Environment Manager that launches the agent. DS module provides name, resource, and public credential to location and network identity mappings. RM is the gatekeeper for the platform and decides if the platform has sufficient resources available to accept the execution of an agent. RM also manages access controls for AEE "containers." Finally, RM monitors use of resources and either warns or terminates misbehaving agents. The CN module is responsible for reliable and secure transfer of packaged agents and peer-to-peer communication between VOLTTRON platforms.

VOLTTRON is currently implemented in Python v2.7 and leverages many existing Python modules developed by the open source community.

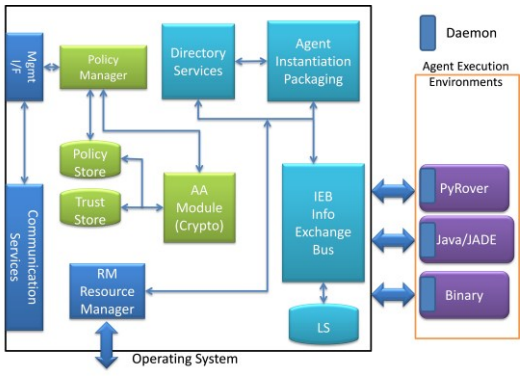

Figure 1. Volttron platform components

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

Figure 2. Volttron Demonstration testbed

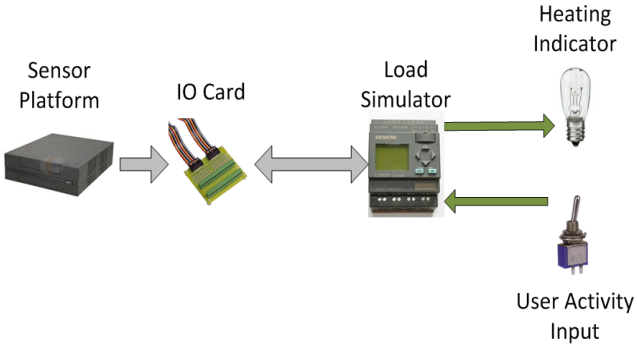

Figure 3. Volttron responsive asset control

3. Demonstration Hardware and Software

In order to test and improve VOLTTRON™ in a realistic environment, we built a hardware demonstration. In an example, as we will describe in Section 3.2, we demonstrate agents, running in our platform, working with each other to control appliances and keep the "neighborhood" under an energy goal.

3.1 Description of Hardware

We have four "homes" in our demonstration that consist of an embedded computer connected to the physical world via digital and analog IO cards. These IO cards provide TTL digital signals to a level conversion board allowing the interface with a stand Programmable Logic Controllers (PLC). Water heater and electric vehicle state behavior is emulated by a PLC and based on a thermal dynamic model of a water heater and real data collected from a Nissan Leaf. The use of a PLC for load modeling decouples this work from the development of the Volttron platform and allows for standalone testing and validation. The testbed is shown in Figure 2. The system shown in Figure 2 is illustrated as a block diagram in Figure 3.

3.2 Demonstration Agents

The agent system implemented for the demonstration is based on the system we described in [1,2]. This demonstration focuses on keeping energy usage for the "neighborhood" below some maximum threshold. To achieve the conservation goal, household agents monitor the energy usage of appliances and broadcast energy requirements and priorities to other participating agents. The hardware portion of the demonstration is meant to reflect current appliances and shares their limitations. The water heater cannot report heat level, only whether a bottom or top element is drawing power. The sole input into the water heater is a conserve signal which prevents it from drawing power. However, it will ignore this signal if the top element is running since that indicates low energy in the tank and could lead to reduced quality of service for the homeowner.

The outputs for the electric vehicle (EV) are whether it is charging, completely charged, or driving. Similar to the conserve signal on the water heater, the EV can be told not to charge.

Priorities were set for the different components based on maintaining a reasonable quality of service for the homeowner to prevent opt-out. The bottom element of the water heater has lowest priority 1, the EV 3, and the top element of the water heater 5. Due to the override, the top element will use power regardless of a conserve signal and thus needs the highest priority to prevent going over load.

Table 1. Example scenario details

	1	2	3	4	5	6	7	8	9
Car1	Idle	Idle	ON	Wait	ON	Idle	Idle	Idle	Idle
Car2	Idle	Idle	Idle	Wait	Wait	ON	Idle	Idle	Idle
WH1	Idle	BOT	Wait	TOP	Wait	Wait	BOT	Idle	Idle
WH2	Idle	Wait	Wait	TOP	Wait	Wait	Wait	BOT	Idle

In an example scenario, we start with all appliances fully charged and not drawing power. Then residents begin taking morning showers thus depleting hot water. This is simulated by a user activity input switch on the hardware as shown in Figures 2 and 3. When the water temperature falls below a preset point the bottom elements turn on to reheat the water. The agents in charge of the water heaters then broadcast their energy needs along with their priority. Appliances with the same priority are assigned a random tie breaker. In this case the water heater for household WH1 wins the tie and therefore its bottom element may draw power. In the demonstration this is indicated by a bulb in Figure 3. Next the car for household Car1 returns and needs to charge. Since car charging has priority 3 it takes precedence over WH1's water heater. While the car charges, both water heaters continue to lose energy as their residents shower. Eventually, the energy drops low enough that the top elements kick on at the highest priority causing the car to now wait. Car2 then returns home and must also wait for its chance to charge. Once the water heaters are recharged to some minimum, their top elements turn off. Car1 then takes priority with a higher tie breaker than Car2. Once Car1 finishes charging, Car2 gets priority. When that finishes, the water heaters then get a chance to run their bottom elements with WH1 running to completion followed by WH2. The households are then back at their fully charged states.

4. REFERENCES

[1] Akyol, B., Haack, J, Tews, C., Carpenter, B., Kulkarni, A., and Craig, P.. 2011. An Intelligent Sensor Framework for the Power Grid. *In ASME Conf. Proc. 2011, 1485 (2011),* DOI=10.1115/ES2011-54619

[2] Akyol, B., Haack, J., Ciraci, S., Carpenter, B., Vlachopoulou, M. and Tews, C. 2012. VOLTTRON: An Agent Execution Platform for the Electric Power System. In *Proceedings of the 3rd International Workshop on Agent Technologies for Energy Systems* (Valencia, Spain, June 5, 2012).