Volttron Implementation: Automated Fault Detection and Diagnosis for AHU-VAV Systems

Volttron Workshop

23 July 2015
Arlington, VA

Adam Regnier
Jin Wen, Ph.D.

Building Science & Engineering Group
Drexel University
Philadelphia, PA
Overview/Agenda

1. Introduction
   - Faults in AHU-VAV Systems

2. Diagnostics
   - Methods & Requirements

3. Volttron Integration
   - Platform overview
   - Drivers
   - Database
   - Agents

4. Why Volttron?
   - Benefits for research & for in

5. Going Forward
AHU-VAV Systems

AHUs are utilized in over 30% of all commercial building floor space

- Air handling units (AHUs) manage heat/energy exchange and ventilation
- As a result, faulty operation has significant energy and health/safety impacts

Difficulties for AHU-VAV systems

- “Built-up” (custom) one-of-a-kind systems
- Low sensor density and quality
- Multiple operational modes
- Continuously transient operation
- Non-linear system
Diagnostics

**Market-driven challenges for Automated Fault Detection and Diagnosis (AFDD)**

- Lack of willingness to invest in AFDD
  - Difficult to demonstrate the value
- Physical system upgrades
- Engineering time/expense for method refinement increases costs
- Low tolerance for false alarms
- Require a non-intrusive strategy that will not impact:
  - Control strategies
  - Comfort
Faults

Mechanical Failure/Degradation
- Dampers
- Valves
- Fans
- Sensors

Control Issues
- Improper Sequencing
- Instability
- Cycling

Maintenance Items
- Filters
- Belts
- Coils

Operator/Maintenance Error
- Forgotten overrides
- Disabled features
- Improper installation/repair
## Value

### Benefits
- Improved comfort (productivity)
- Energy Savings
- Improvement in air quality
- Increased equipment lifetime
- Improved service scheduling

### Costs
- Engineering labor
  - Getting data out of the building
  - Mapping points
  - Customization and tuning
- Additional hardware

**Difficult to accurately quantify!**

**Opportunity to reduce costs through increased automation**
Approach – AHU Diagnostics

Plug-and-play implementation
• Minimal upfront engineering costs (no modeling/customization requirements)
• No specialized training data
• Automatically “learns” system characteristics
• Adapts to any sensor set or configuration

De-couples detection and diagnostic algorithms
• Reduced computational requirements
• Cross-validation of results

Demonstrated to be effective for all types of faults
• Dampers, valves, fans, sensors, controls, etc.

Passive method (no intrusive testing)

Demonstrated using data from four commercial buildings
• Naturally occurring faults and artificial fault experiments
Approach – AHU Diagnostics

AHU

Operational data

Diagnostic BN

PM-PCA Fault Detection

Energy Impact Analysis

Retro-Cx BN

Repair to Fault-Free State

Historical Data

User
Volttron Tasks

AHU

Operational data

Retro-Cx BN

Repair to Fault-Free State

Diagnostic BN

PM-PCA Fault Detection

Energy Impact Analysis

Historical Data

User
Volttron Demonstration

Stratton Hall (Philadelphia, PA)
- Three stories, 74,000 ft²
- Mix of offices and classrooms
- Psychology department building
- 2 AHUs, 54 VAV-boxes
- District chiller cooling
- District steam heating
Volttron Overview

**Volttron Setup**
- Runs on Linux
- Straightforward installation and configuration

**Architecture**
- Data from the building is passed into the message bus
- From the message bus, different agents can act on the data and send control signals to the building using the actuator agent.

**Agent Development**
- Agent development is straightforward based upon the sample agents developed by PNNL
- All of the different components interact using a simple publish/subscribe method
Getting Data

**BACnet Driver**
- Configuration tool automatically detects BACnet traffic on network
- Provides a list of available BACnet point names
- BACnet driver gathers data from the building in real time and publishes to the message bus
Getting Data

Alternate Configuration

• As soon as we configured our Volttron instance using the BACnet driver, our ability to access the data was removed due to security upgrades.

• At our demonstration site, the network configuration was modified so access to the BACnet data required physical relocation of our Volttron computer.

• Since we are doing development, physical access to the Volttron machine is required.

• Wrote a driver to pull data from the BAS server instead (via https API).

Testing and development of this approach are ongoing.
“Active” AFDD

- Active AFDD refers to manipulating the building controls through a set of actions designed to identify faulty behavior.
- There are instances where it is difficult to differentiate between faults with high confidence using passive methods.
- We were interested in testing PNNL’s active AFDD agent, and possibly incorporating it into our work.
- However, we will have to write our own actuator agent due to the same issues presented with the BACnet driver.
“Active” AFDD

- Active AFDD refers to manipulating the building controls through a set of actions designed to identify faulty behavior.
- There are instances where it is difficult to differentiate between faults with high confidence using passive methods.
- We were interested in testing PNNL’s active AFDD agent, and possibly incorporating it into our work.
- However, we will have to write our own actuator agent due to the same issues presented with the BACnet driver.

Presently investigating this option…
Data Historian

sMAP Database

- Presently the only archiving option for Volttron
- Quick for time-series data
- Running on Ubuntu server
- Does not accept string data
- Easy configuration using publish
- Agents can access historical data through the message bus
- Stores results from AFDD agent
- Starting testing as part of a back-end for a web-based AFDD interface this summer

Prior to Volttron being implemented, we had been storing building data on MySQL and “flat databases” optimized for our AFDD algorithm.
AFDD Configuration

Implementation Alternatives (Passive vs. Active)

Initial implementation using a *cloud agent* design connected to a building

Using Matlab for algorithm development speeds agent development and rapid prototyping/testing of algorithms.
AFDD Configuration

Implementation Alternatives (Passive vs. Active)

Implementation using an *active agent* design connected to a building.

Using Matlab for algorithm development speeds agent development and rapid prototyping/testing of algorithms.
Why Volttron?

Decision to Develop a Volttron Agent

- Open-source
- Many platform services already developed and available
  - Database, archiving, drivers for data retrieval
- Integration with other “agents” via a single point of contact for device interface
- Security
- Implementation flexibility
  - Cloud-based application (API)
  - Local application (Volttron control agent for “active” diagnostics)

Allows us to generate a comprehensive solution in less time, and at a lower cost
Implementation Process

Our Process

- Install sMAP server
- Add static building data to sMAP for testing
- Install & test Volttron
- Develop Volttron agent & test data passing
- Connect to Drexel buildings
  - BACnet driver
  - Custom driver

Future Steps

- Continue AFDD demonstration
- Develop web-based AFDD interface
- Integrate active diagnostics
- Add more buildings to our demonstration