

Use of Volttron Platform for Advanced Control of Building Equipment

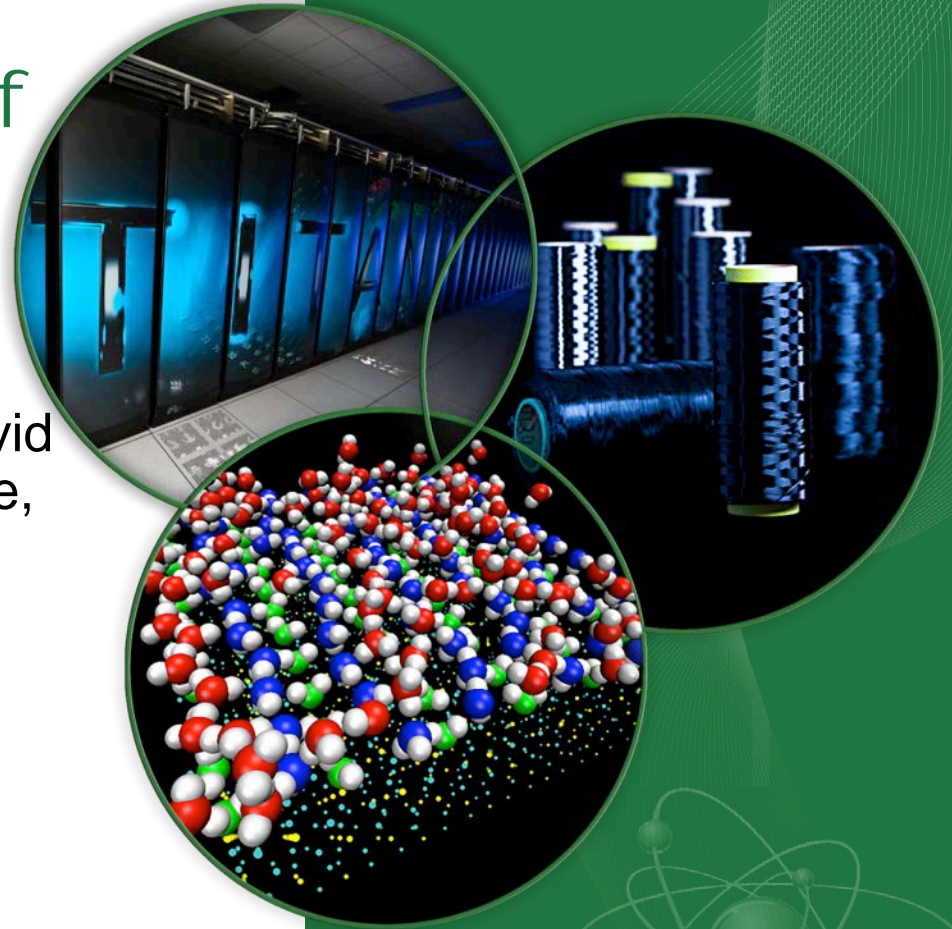
Teja Kuruganti, James Nutaro, David Fugate, Michael Starke, Brian Fricke, Ed Vineyard, Patrick Hughes

Presented to:

Technical Meeting: Software Framework for Transactive Energy

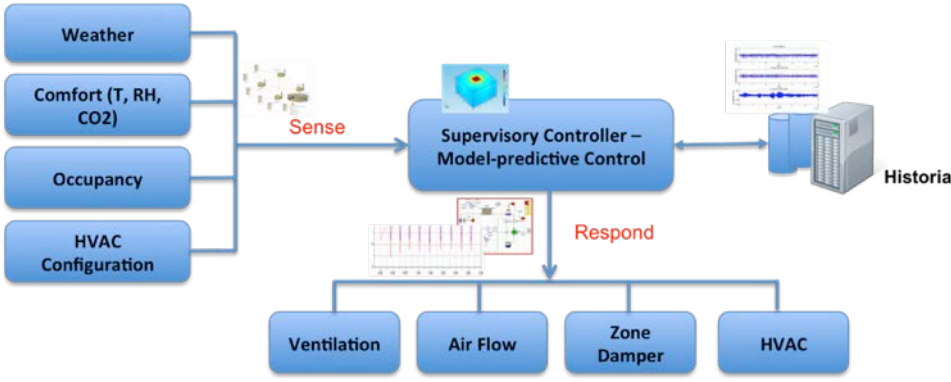
July 23-24, 2014

Case Western Reserve University

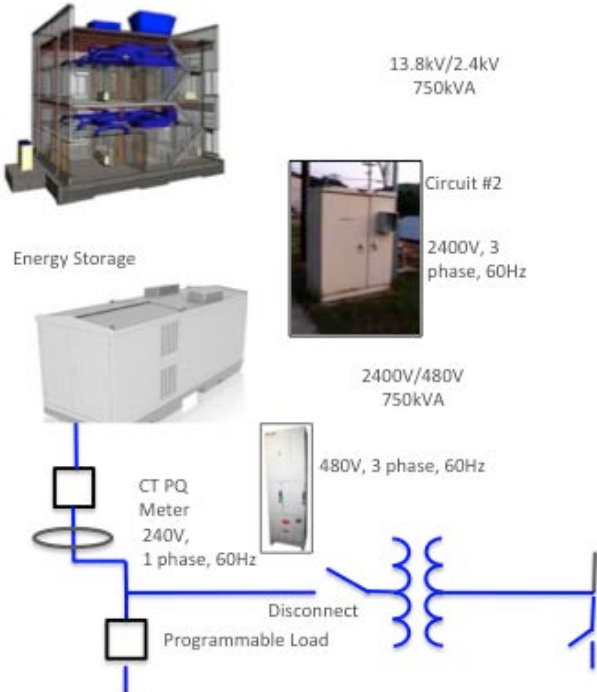


ORNL Transactional Network Applications

Autonomous Control Build control formulation to orchestrate multiple RTUs with in a single building for a particular grid service (peak reduction, renewable integration) and energy efficiency applications (occupancy, weather forecast)



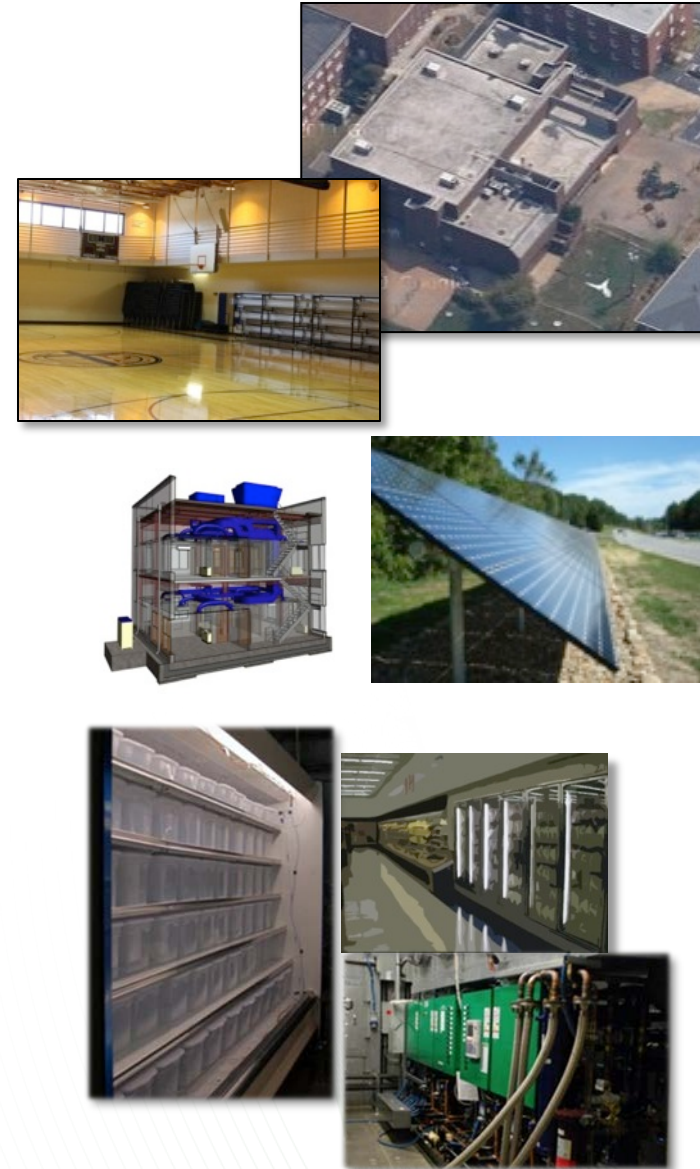
Super-Market Refrigeration Develop applications (in collaboration with Emerson) to utilize refrigeration systems to provide energy services to grid and improve the energy efficiency of these systems



Renewable Integration Build autonomous controller to temporally match RTU energy consumption and peak PV generation using forecasting tools

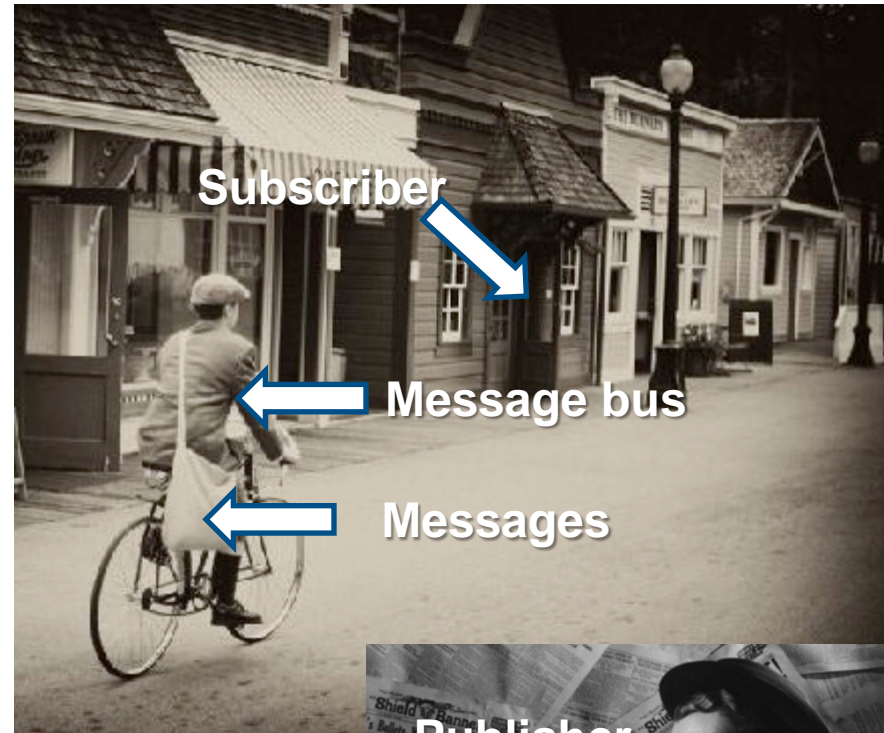
Application Focus

- Develop **control techniques** for reducing peak demand and improve energy efficiency of rooftop units and supermarket refrigeration systems and integrate photovoltaic sources
- Low-cost, **“low-touch” retrofit** of control technology into buildings and refrigeration systems to facilitate transactive opportunities for energy efficiency and with the electric grid
- Demonstrate applications using Volttron platform



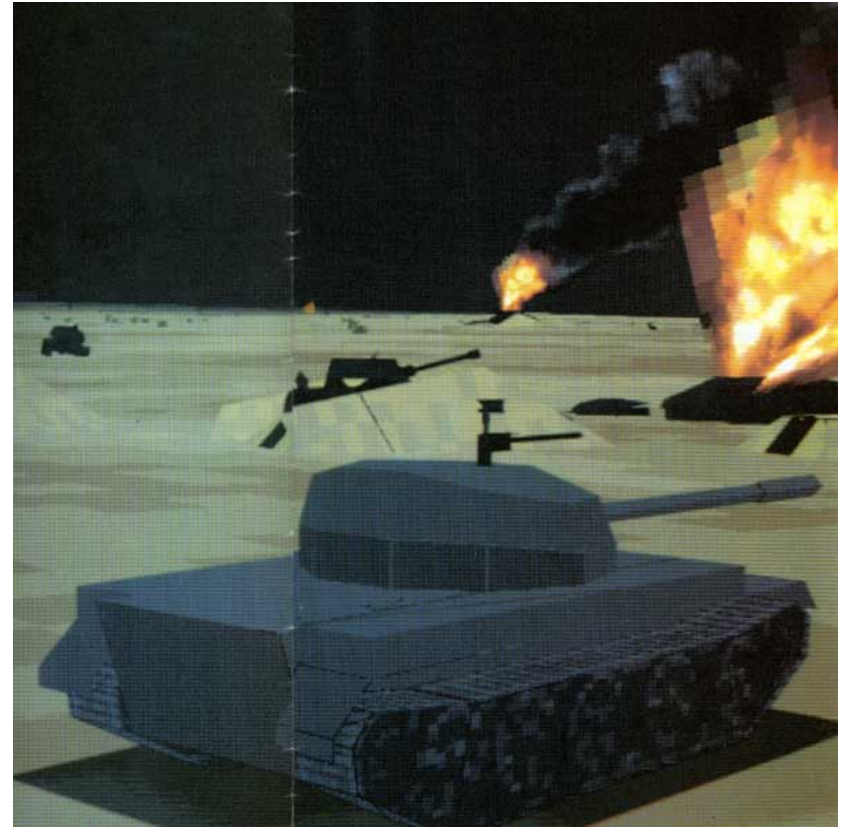
Basic Assumption of Pub/Sub systems

- Intermittent and irregular updates to data subscriptions are consistent with the component's proper operation
 - You can wait for the data
 - You can do without the data
 - Or both



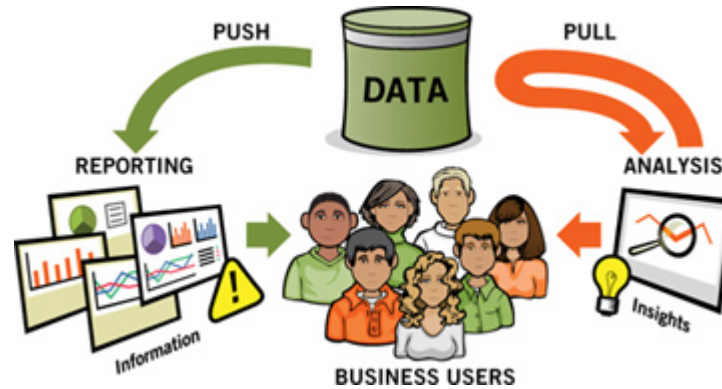
Common types of applications

- Distributed simulation
 - e.g., to train fighter pilots & tank crews
 - Online computer games
- Internet alerting systems
 - Stock prices feeds
 - Rich Site Summary (RSS)
- Data logging systems
 - Collecting data from sensor networks
 - On-line diagnostic systems



Simnet screenshot From Bruce Sterling's "War is Virtual Hell," Wired 1.01 (March-Apr. 1993)

Push vs. Pull



- A publish subscribe system is a “push” system
 - Data is pushed to you by a publisher whenever that publisher has data to send
 - Publisher determines what you see and when
 - e.g., the publisher may not report errors with a sensor
- Polling is an example of a “pull” system
 - You ask for data when you need it
 - You get what you ask for
 - e.g., you can check for errors values following a read

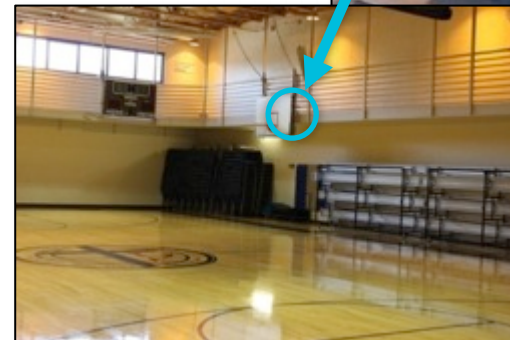
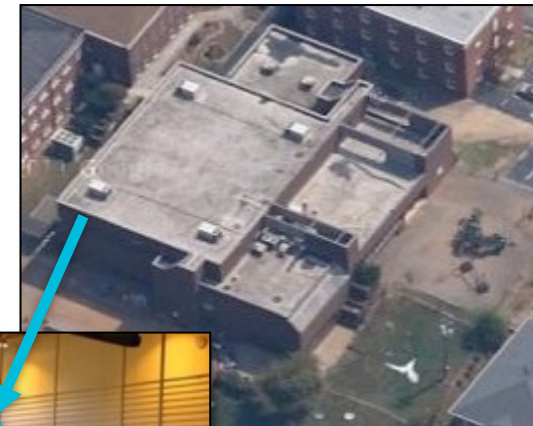
Illustration: Autonomous Control App

Problem

- Each HVAC unit and thermostat act as an isolated pair
 - All HVAC units tend to run at the same time
 - High peak demand
- No accounting for natural heat exchange
 - Some overheating and overcooling
- But this does maintain a comfortable temperature and so occupants are happy
- These are not problems for the occupants
- Just for the building owner who pays the bills

Solution

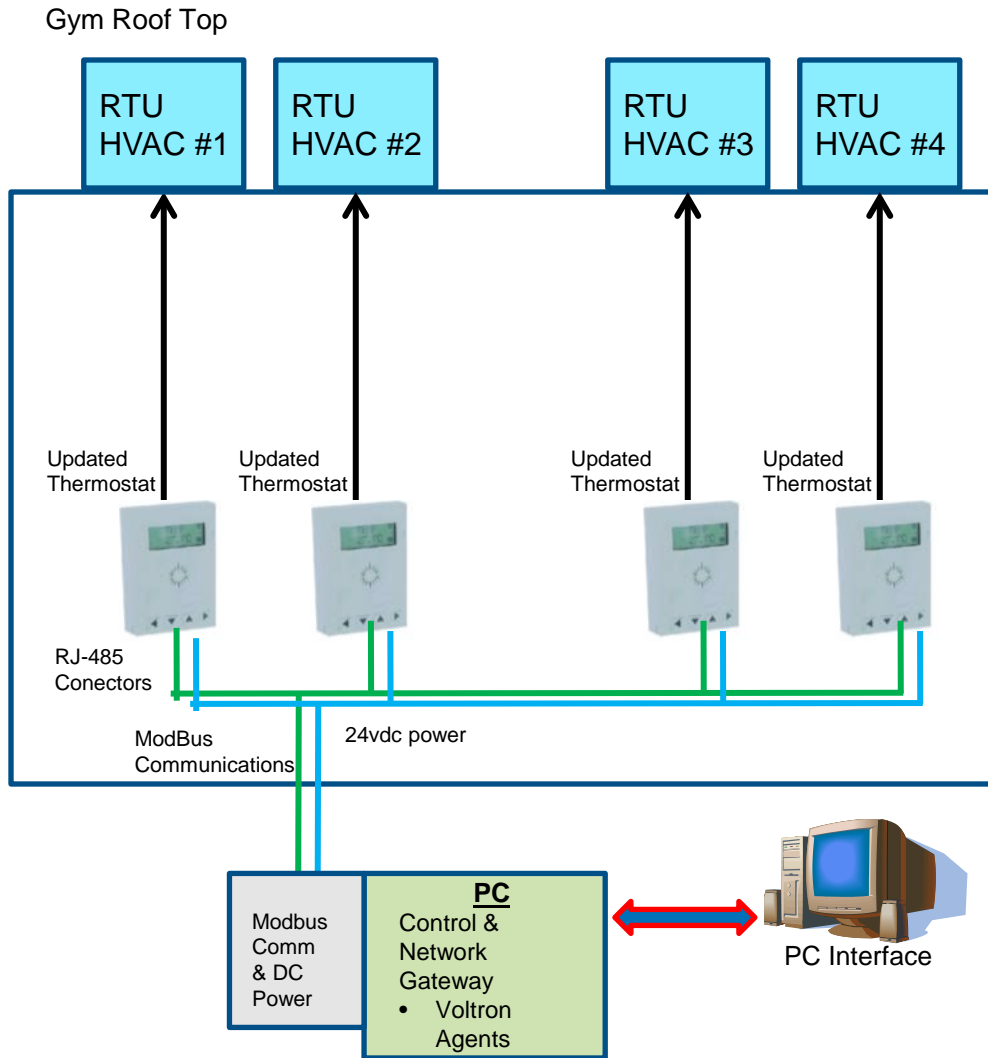
- Sequence the HVAC units to avoid running all at the same time
 - Reduce peak demand
- Account for “free” heating and cooling due to natural heat exchange with outside air and between zones
 - Avoid overheating overcooling
- Do this in a way that maintains comfort



Autonomous Control Application

- Pulls data from the building thermostats
 - We need current data at each control interval
 - We want to know if there are problems with the temperature sensor
 - Data is obtained directly from the thermostat
- Subscribes to outdoor temperature data
 - This is supplemental to the control, improving it but not essential for operation
 - Obtained via a web service

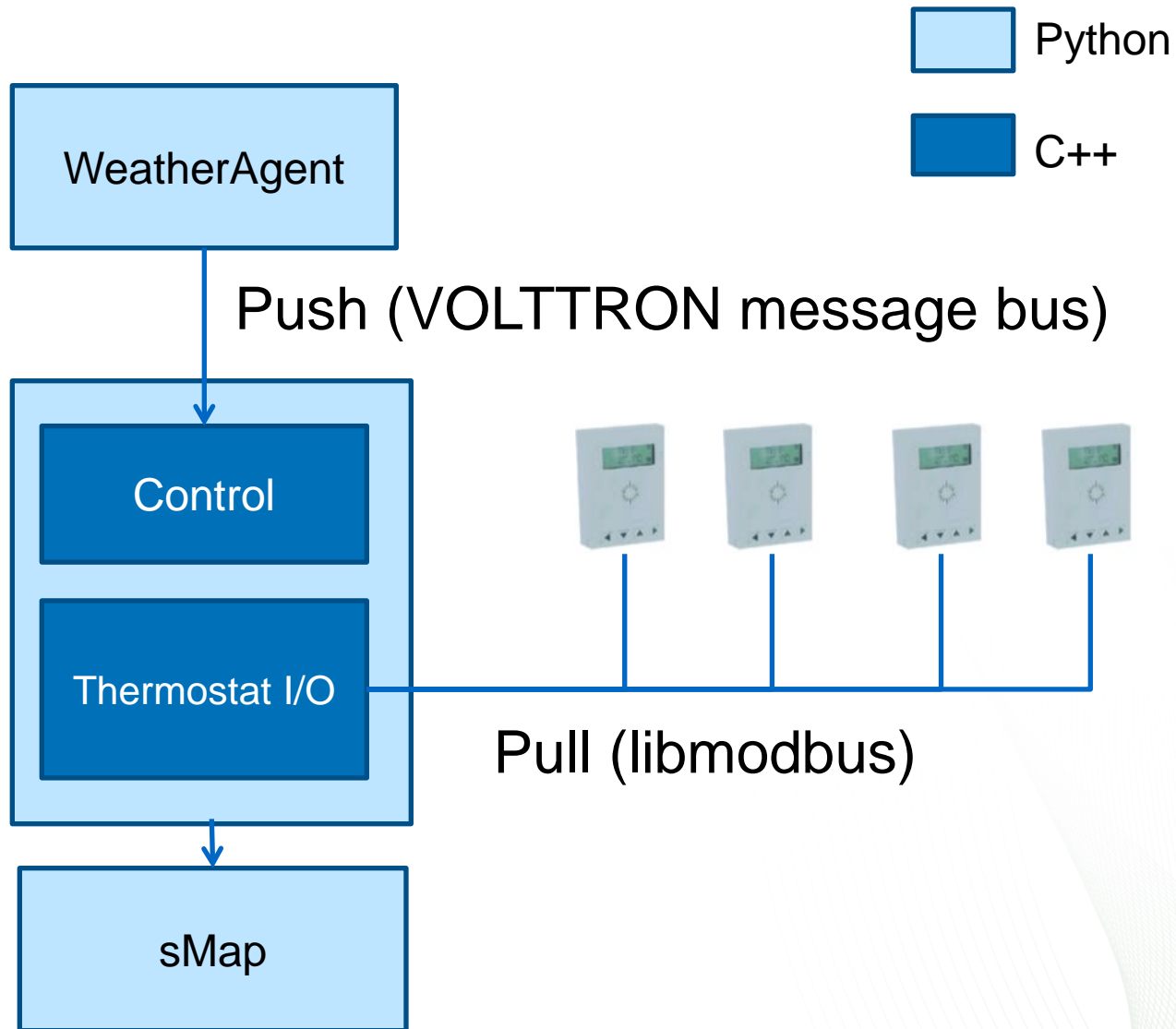
Schematic of Deployed System



Family Life Center Gym Deployment

- Wired ModBus network to the thermostats
- Weather data obtained through the Internet
- Control software resides in a PC at the deployment site

Software components



“Push” style in a VOLTTRON Agent

```
class MpcAgent(PublishMixin, BaseAgent):
    def __init__(self, config_path, **kwargs):
        super(MpcAgent, self).__init__(**kwargs)
        self.config = utils.load_config(config_path)
        self.mpc = MPC.MPC()
        self.mpc.make_gui()
    def setup(self):
        self._agent_id = self.config['agentid']
        super(MpcAgent, self).setup()
    @matching.match_exact('weather/temperature/temp_f')
    def on_match(self, topic, headers, message, match):
        self.mpc.set_outdoor_temp(float(message[0]))
    @periodic(MPC.PERIOD)
    def run_control(self):
        self.mpc.run_control(MPC.PERIOD)
```

VOLTTRON Agent

```
class MpcAgent(PublishMixin, BaseAgent):  
    def __init__(self, config_path, **kwargs):  
        super(MpcAgent, self).__init__(**kwargs)  
        self.config = utils.load_config(config_path)  
        self.mpc = MPC.MPC()  
        self.mpc.make_gui()  
    def setup(self):  
        self._agent_id = self.config['agentid']  
        super(MpcAgent, self).setup()  
    @matching.match_exact('weather/temperature/temp_f')  
    def on_match(self, topic, headers, message, match):  
        self.mpc.set_outdoor_temp(float(message[0]))  
    @periodic(MPC.PERIOD)  
    def run_control(self):  
        self.mpc.run_control(MPC.PERIOD)
```

Initialize the VOLTTRON base class and load VOLTTRON specific configuration information

VOLTRON Agent

```
class MpcAgent(PublishMixin, BaseAgent):
    def __init__(self, config_path, **kwargs):
        super(MpcAgent, self).__init__(**kwargs)
        self.config = utils.load_config(config_path)
        self.mpc = MPC.MPC()
        self.mpc.make_gui()
    def setup(self):
        self._agent_id = self.config['agentid']
        super(MpcAgent, self).setup()
    @matching.match_exact('weather/temperature/temp_f')
    def on_match(self, topic, headers, message, match):
        self.mpc.set_outdoor_temp(float(message[0]))
    @periodic(MPC.PERIOD)
    def run_control(self):
        self.mpc.run_control(MPC.PERIOD)
```

Create Python classes that wrap the C++ code for the control logic and for interacting with the thermostats

VOLTRON Agent

```
class MpcAgent(PublishMixin, BaseAgent):
    def __init__(self, config_path, **kwargs):
        super(MpcAgent, self).__init__(**kwargs)
        self.config = utils.load_config(config_path)
        self.mpc = MPC.MPC()
        self.mpc.make_gui()
    def setup(self):
        self._agent_id = self.config['agentid']
        super(MpcAgent, self).setup()
    @matching.match_exact('weather/temperature/temp_f')
    def on_match(self, topic, headers, message, match):
        self.mpc.set_outdoor_temp(float(message[0]))
    @periodic(MPC.PERIOD)
    def run_control(self):
        self.mpc.run_control(MPC.PERIOD)
```

Receive updates to the temperature data and supply these to the control module

VOLTRON Agent

```
class MpcAgent(PublishMixin, BaseAgent):
    def __init__(self, config_path, **kwargs):
        super(MpcAgent, self).__init__(**kwargs)
        self.config = utils.load_config(config_path)
        self.mpc = MPC.MPC()
        self.mpc.make_gui()
    def setup(self):
        self._agent_id = self.config['agentid']
        super(MpcAgent, self).setup()
    @matching.match_exact('weather/temperature/temp_f')
    def on_match(self, topic, headers, message, match):
        self.mpc.set_outdoor_temp(float(message[0]))
    @periodic(MPC.PERIOD)
    def run_control(self):
        self.mpc.run_control(MPC.PERIOD)
```

Execute the control logic every 10 minutes. This polls the thermostat temperature, determines which RTUs to run, and then switches the thermostat relays accordingly.

"Pull" style w/o VOLTTRON

```
class MPC:
    def __init__(self):
        self.bldg = python_building.Building()
        self.cntrl = python_control.Control(self.bldg.get_num_zones())
    def set_outdoor_temp(self, degF):
        self.bldg.set_outdoor_temp(degF)
    def get_outdoor_temp(self):
        return self.bldg.get_outdoor_temp()
    def run_control(self):
        self.bldg.advance()
        for zone in range(0, self.bldg.get_num_zones()):

self.cntrl.set_upper_limit(zone, self.bldg.get_high_temp_limit(zone))

self.cntrl.set_lower_limit(zone, self.bldg.get_low_temp_limit(zone))
    self.cntrl.set_zone_temp(zone, self.bldg.get_indoor_temp(zone))
    self.cntrl.set_outside_temp(self.bldg.get_outdoor_temp())

    self.cntrl.run_control()
    for zone in range(0, self.bldg.get_num_zones()):
        self.bldg.set_hvac_mode(zone, self.cntrl.get_hvac_command(zone))

    def get_control_period(self):
        return self.cntrl.get_control_period()
```


Pulling temperature

```
def get_temp():  
    f =  
urllib2.urlopen('http://api.wunderground.com/.../Knoxville.json')  
    json_string = f.read()  
    parsed_json = json.loads(json_string)  
    location = parsed_json['location']['city']  
    temp_f = parsed_json['current_observation']['temp_f']  
    return temp_f
```

Main loop

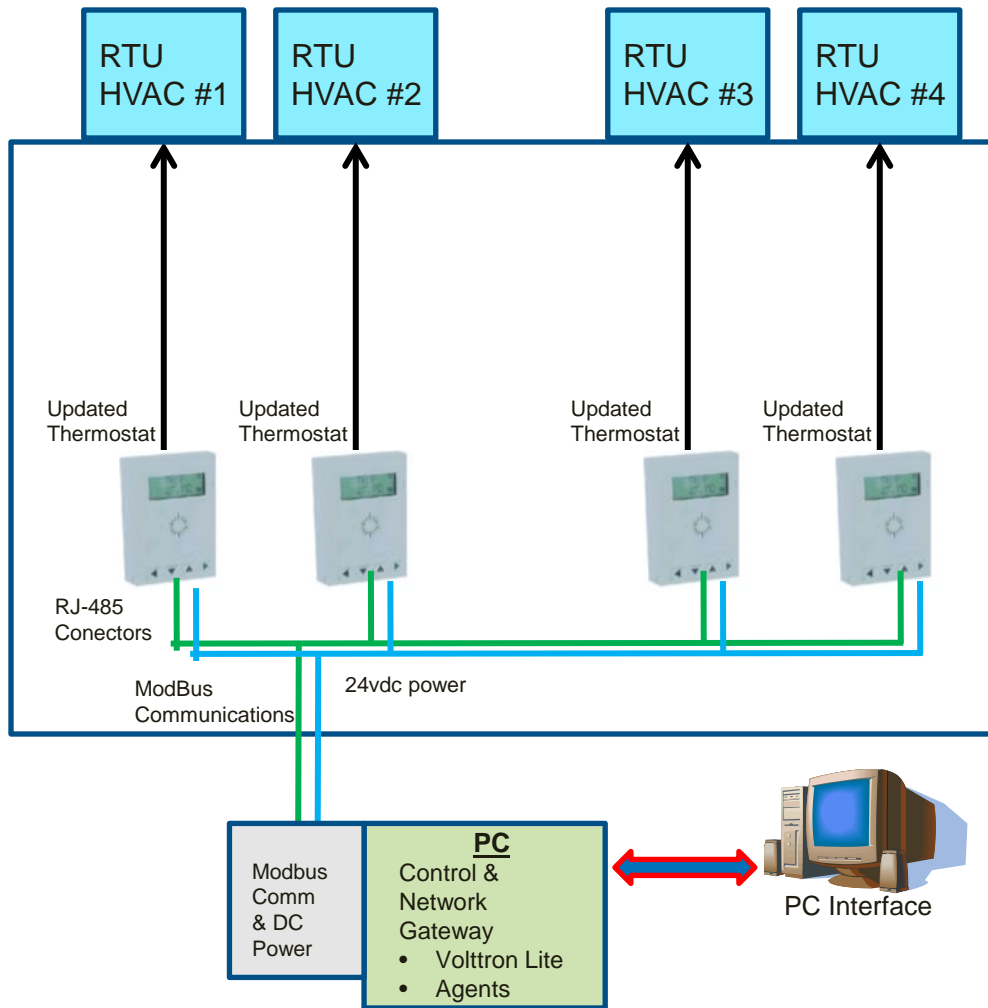
```
def main():  
    mpc = MPC()  
    temp_f = 25  
    while True:  
        try:  
            temp_f = get_temp()  
        except:  
            print "Could not get weather data"  
        mpc.set_outdoor_temp(temp_f)  
        period = mpc.get_control_period()  
        mpc.run_control()  
        time.sleep(period)
```

SLOC by module*

- VOLTRON Python code for agent: 34
- Python code for standalone version: 62
- Code for MPC module (same for both versions)
 - Python wrapper: 55 Python, 115 C++
 - C++ code: 484
- Code for thermostat module (same for both versions)
 - Python wrapper: 68 Python, 14 C++
 - C++ code: 404

* Calculated with David Wheeler's sloccount

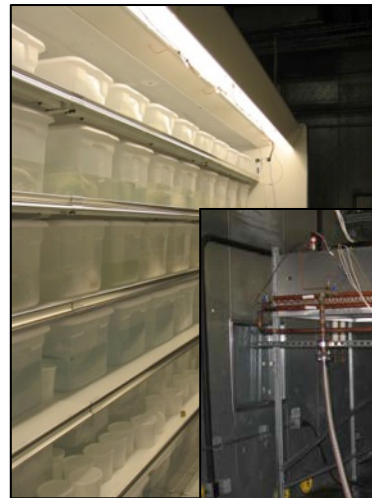
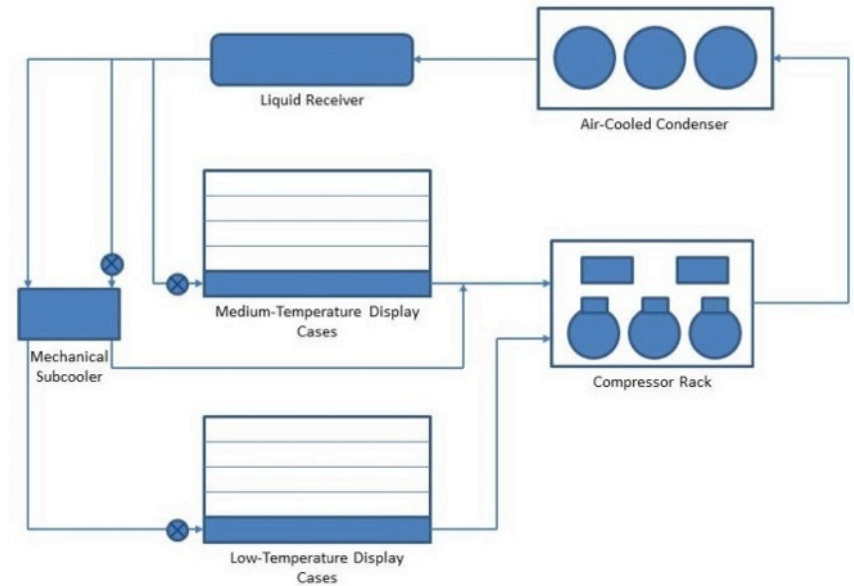
Deployment Architecture #1



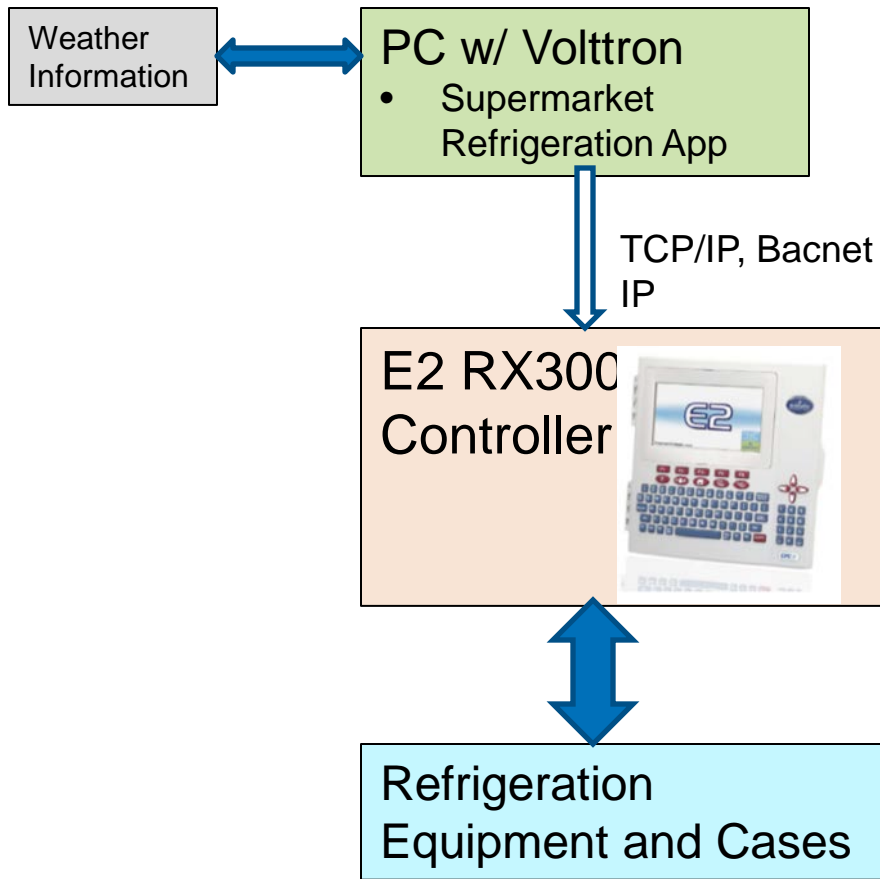
- Replace existing thermostats with MODBUS thermostats.
- Cable for MODBUS
- Deploy PC with VOLTTRON

ORNL Supermarket Refrigeration System

- Components
 - Compressor Rack
 - Air-cooled Condenser
 - Medium-temperature (MT) display cases
 - Low-temperature (LT) display cases
- Refrigerating Capacity
 - LT: 5 tons at -20°F
 - MT: 10 to 15 tons at 25°F
- LT Load:
 - 3 open vertical display cases (12 ft each)
- MT Load:
 - 2 open vertical display cases (12 ft each)
 - “False” load provided by a plate heat exchanger and glycol loop
- Equipment installed in two temperature and humidity controlled environmental chambers



Deployment Architecture #2

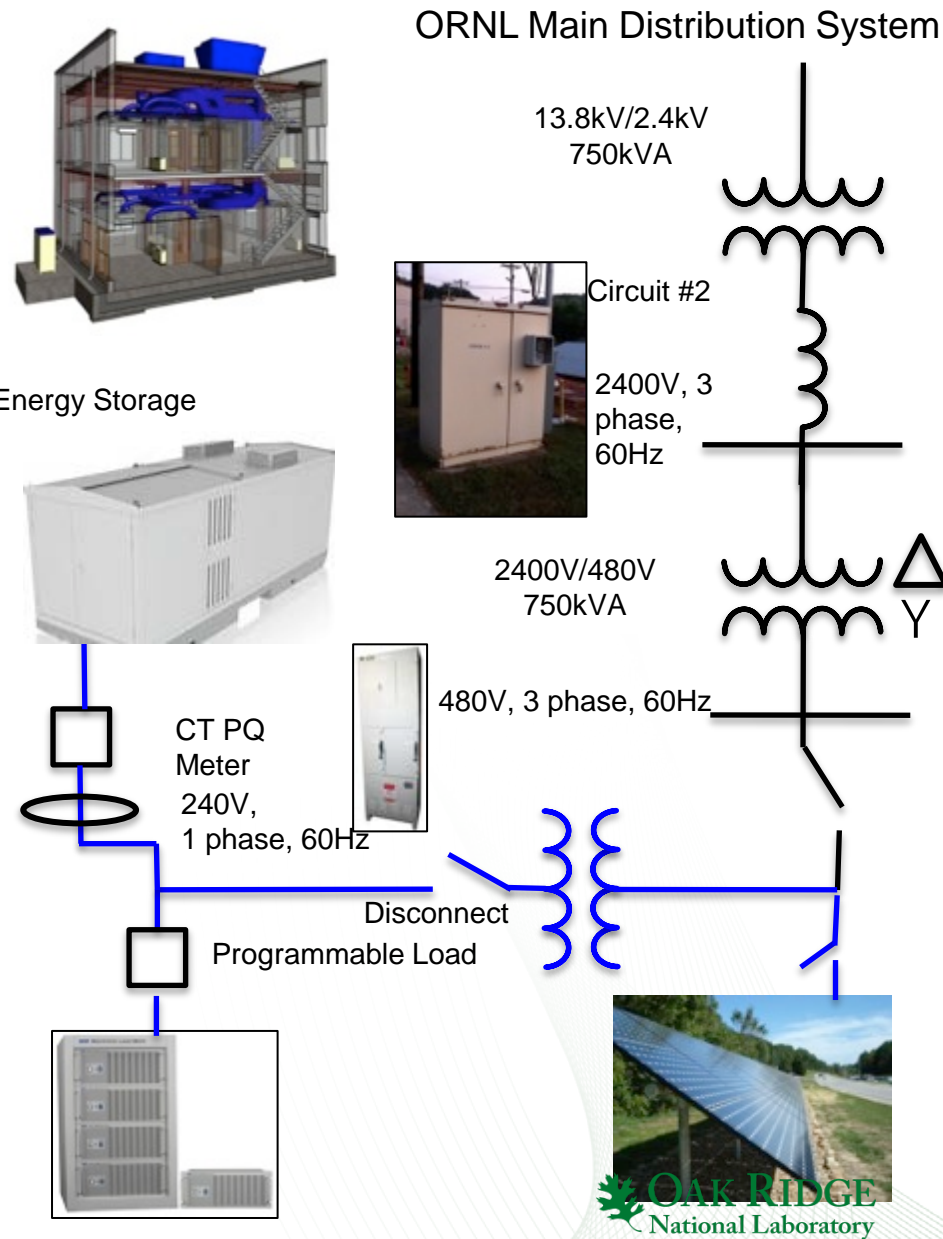


Volttron Applications Identified

1. Smart Defrost
 - A. Demand Response
 - B. Intelligent Defrost (sensing, algorithms, etc.)
2. Peak Reduction
 - A. Coordination of Defrost and operation of all cases.
 - B. Capacity Modulation for Peak Reduction
3. Equipment Prognostics

Renewable Integration Application

- Build controller to temporally match peak energy consumption to PV generation
- Predict variability in PV availability using forecasting tools
- ORNL experimental facilities include:
 - 50kW photo voltaic array
 - 24kW energy storage (battery)
 - 36kW variable load
 - Smart appliances
 - HVAC units (FRPs)

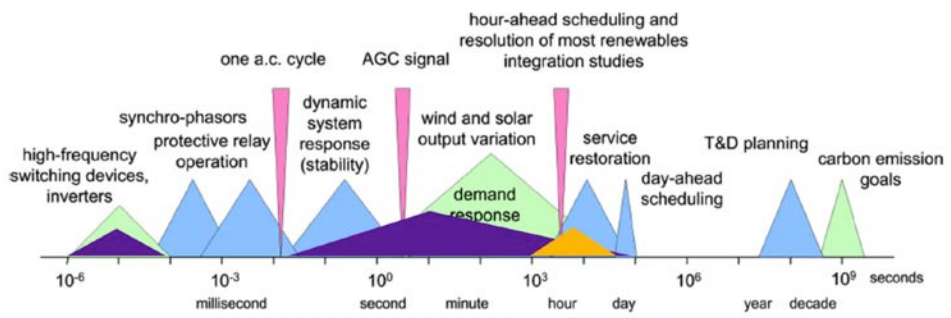
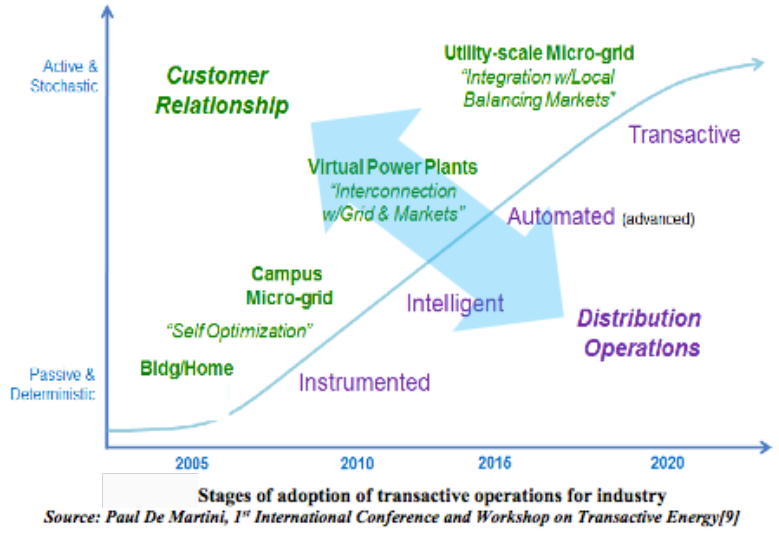


Some observations for our application

- Testing and development of control and actuation/sensing was easiest in a standalone application
 - Simulation-based testing of control functions
 - Bench testing of thermostat modules
 - Integration of weather data, control, and thermostat modules
- Port to VOLTTRON was a useful learning experience, with similar functionality
 - Very small application (about 1,000 SLOC)
 - Good candidate for a “pull” architecture (this was used in the test version)
 - Access to essential data sources (Modbus devices and weather data) are readily supported
 - Repeatable installation of software
 - Addition to library of apps (develop community)
 - New code needed for porting existing interfaces
 - Data exchange specification is not standardized – As applications grow this needs addressing

Future Applications Support Transactive Energy

- Transactive energy requires high-speed wide area control of loosely coupled loads
- Control response can be generated in a centralized or decentralized fashion
 - Utility level information
 - Building-level loads
- Embedded transactive devices that can control building systems over wide-area heterogeneous networks
 - How to guarantee quality of service?



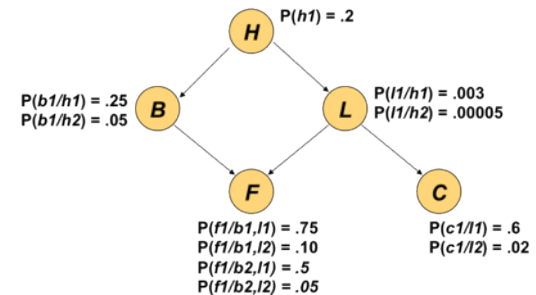
“ To 33% and Beyond: Grid Integration Challenges for Renewable Generation”, Alexandra von Meier, CIEE, presented to UCLA Smart Grid Thought Leadership Forum, March 28, 2012

In-Network Intelligence

- Decentralized **agents** with defined dynamics
- Establish **communication** graph of the network of nodes
- Communications channel constraints
- Develop **strategy** that needs to be executed – Peak Reduction etc.
- Decentralized control execution
 - slow time constant systems
 - fail-safe controls
 - low-commissioning costs

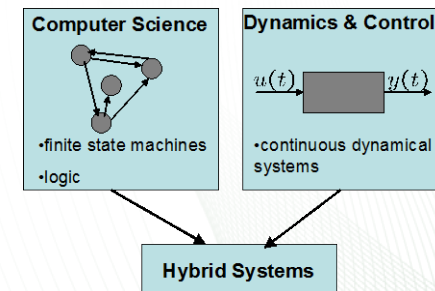
$$\dot{x}^i = f^i(x^i, u^i) \quad x^i \in \mathbb{R}^n, u^i \in \mathbb{R}^m$$

$$y^i = h^i(x^i) \quad y^i \in \mathbb{R}^q$$



$$y_j^i[k] = \gamma y^i(t_k - \tau_j) \quad t_{k+1} - t_k > T_r$$


$$J = \int_0^T L(x, \alpha, \mathcal{E}(t), u) dt + V(x(T), \alpha(T)),$$



Moving Forward

- Applications that are a good fit for implementing with VOLTTRON will have several distinct features:
 1. They naturally call for a publish/subscribe type architecture
 - e.g., applications consisting of large numbers of loosely coupled sub-systems that can be wrapped in an agent
 2. Can make good use of functionality that is part of the VOLTTRON system
 - e.g., coordinating access to shared resources
 3. Are readily conceived as performing tasks that can be accomplished by autonomous, but communicating, agents

Discussion



OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE
FOR U.S. DEPARTMENT OF ENERGY

Recent Results

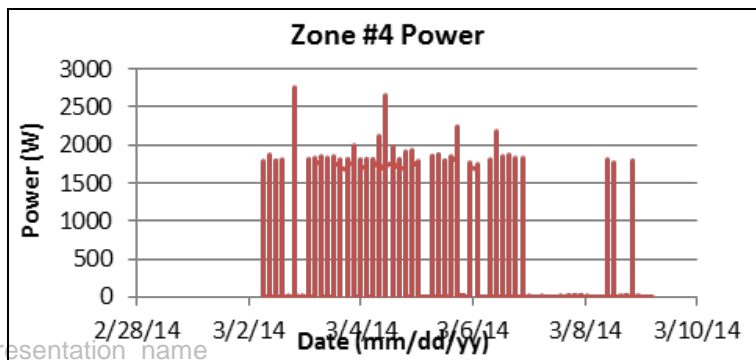
- Deployment Experience and Data

Transaction Network Platform Research for Autonomous Control
Heating, Ventilation, and Air-Conditioning (HVAC) Controls Summary Report

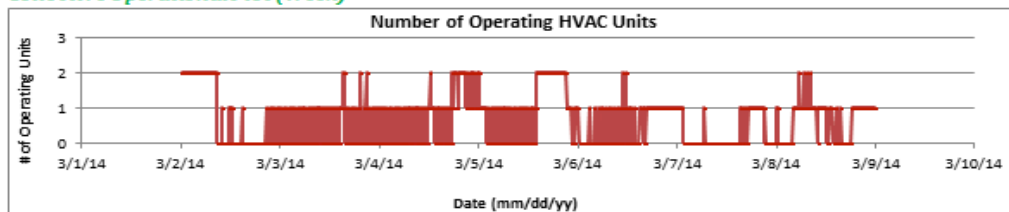
- March 2-8, 2014 Data
 - Control limits number of operating units to not exceed two.

- Setpoint of each zone

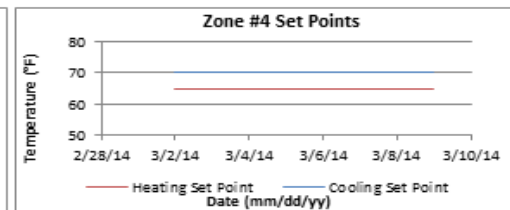
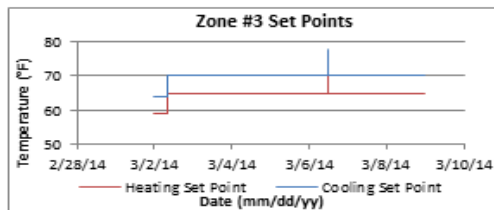
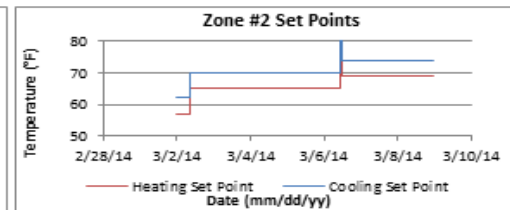
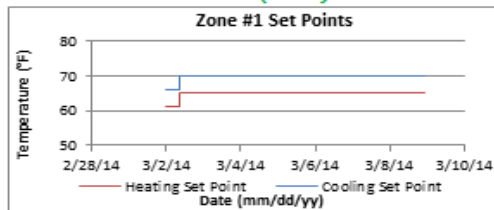
- Submetered power data for each zone



Collective Operational Plot (Week)



Individual Set Point Plots (Week)



Recent Results

- Deployment Experience and Data

**Transaction Network Platform Research for Autonomous Control
Heating, Ventilation, and Air-Conditioning (HVAC) Controls Summary Report**

- March 2-8, 2014 Data
 - Zone temperature versus outdoor temperature

- Operating mode of each zone



**Transaction Network Platform Research for Autonomous Control
Heating, Ventilation, and Air-Conditioning (HVAC) Controls Summary Report**

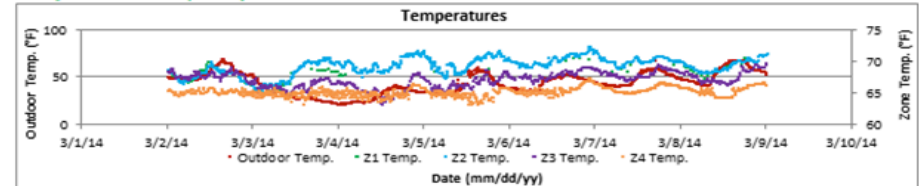
Location: Central Baptist (Fountain City) – Family Life Center
Report Period: March 2, 2014 to March 8, 2014
Number of Roof-Top Units: 4



Temperature Statistics (Week)

	Average	Max.	Min.
Outdoor Temperature (°F):	44.54	69.40	22.90
Zone #1 Temperature (°F):	68.00	71.30	65.00
Zone #2 Temperature (°F):	69.24	72.50	65.50
Zone #3 Temperature (°F):	67.18	69.90	63.30
Zone #4 Temperature (°F):	65.30	67.30	63.00

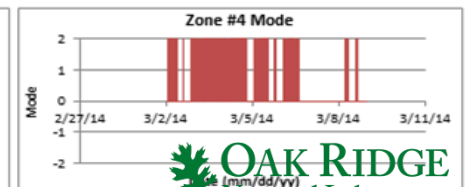
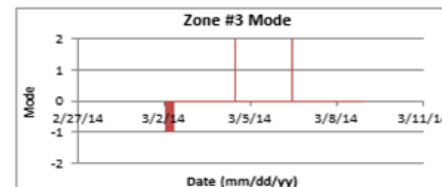
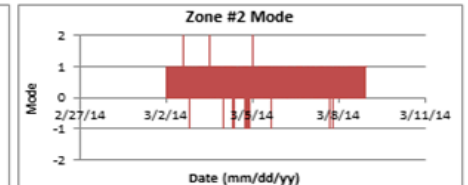
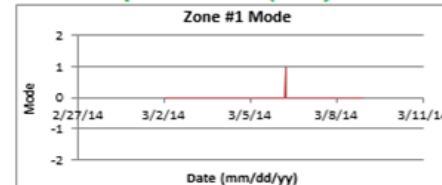
Temperature Plots (Week)



Operational Statistics (Week)

	Zone #1	Zone #2	Zone #3	Zone #4
Two-Stage Cooling (-2):	0.00%	0.00%	0.00%	0.00%
Single-Stage Cooling (-1):	0.00%	0.31%	6.17%	0.00%
Idle (0):	99.99%	55.30%	93.59%	80.82%
Single-Stage Heating (+1):	0.01%	44.39%	0.00%	0.00%
Two-Stage Heating (+2):	0.00%	0.01%	0.24%	19.18%

Individual Operational Plots (Week)



Estimating Summer Energy Savings

- Deployment Experience and Data

Transaction Network Platform Research for Autonomous Control

Heating, Ventilation, and Air-Conditioning (HVAC) Controls Summary Report

- February 23 – March 1, 2014 Data



Transaction Network Platform Research for Autonomous Control Heating, Ventilation, and Air-Conditioning (HVAC) Controls Summary Report

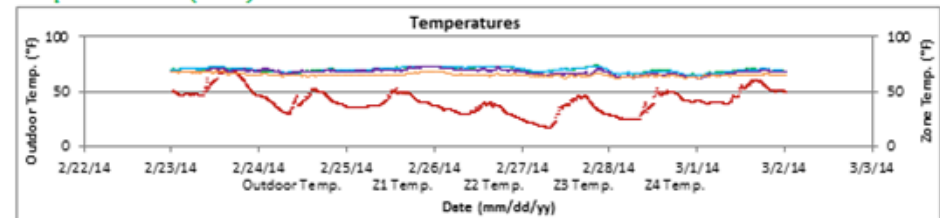
Location: Central Baptist (Fountain City) – Family Life Center
Report Period: February 23 - March 1, 2014
Number of Roof-Top Units: 4



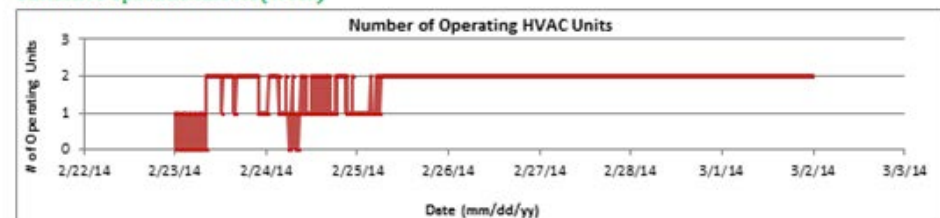
Temperature Statistics (Week)

	Average	Max.	Min.
Outdoor Temperature (°F):	41.45	69.90	17.50
Zone #1 Temperature (°F):	69.96	74.60	65.40
Zone #2 Temperature (°F):	70.30	74.30	64.90
Zone #3 Temperature (°F):	68.35	73.50	62.60
Zone #4 Temperature (°F):	65.93	68.90	61.50

Temperature Plots (Week)



Collective Operational Plot (Week)



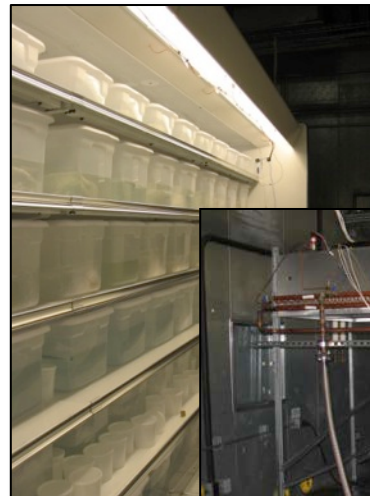
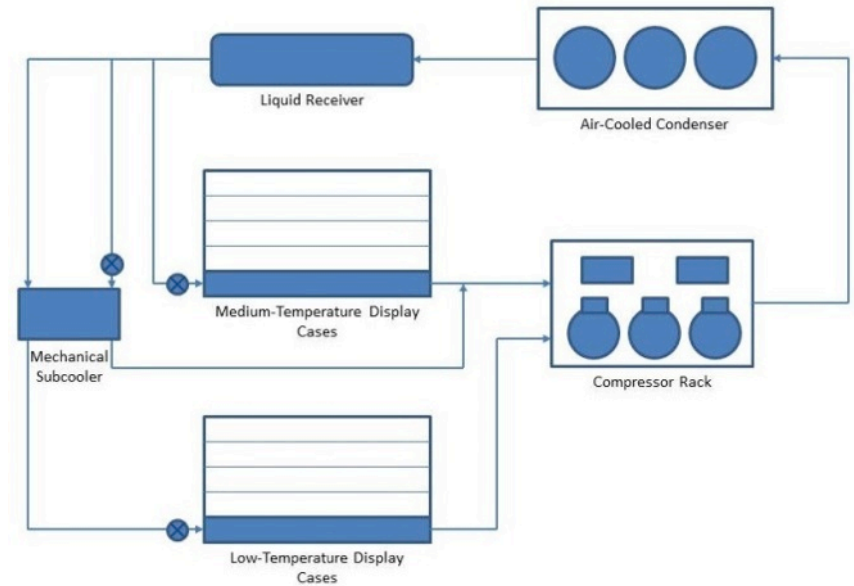
- Zone temperature versus outdoor temperature

- Control limits number of operating units to not exceed two.



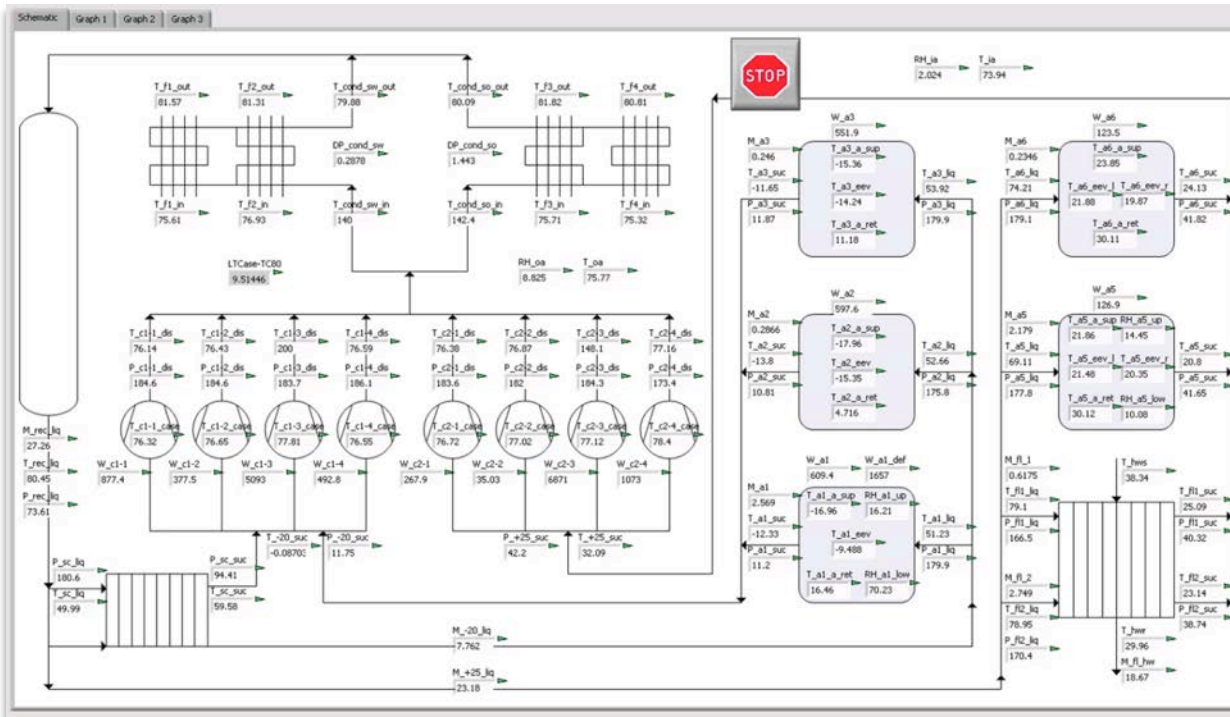
ORNL Supermarket Refrigeration System

- Components
 - Compressor Rack
 - Air-cooled Condenser
 - Medium-temperature (MT) display cases
 - Low-temperature (LT) display cases
- Refrigerating Capacity
 - LT: 5 tons at -20°F
 - MT: 10 to 15 tons at 25°F
- LT Load:
 - 3 open vertical display cases (12 ft each)
- MT Load:
 - 2 open vertical display cases (12 ft each)
 - “False” load provided by a plate heat exchanger and glycol loop
- Equipment installed in two temperature and humidity controlled environmental chambers

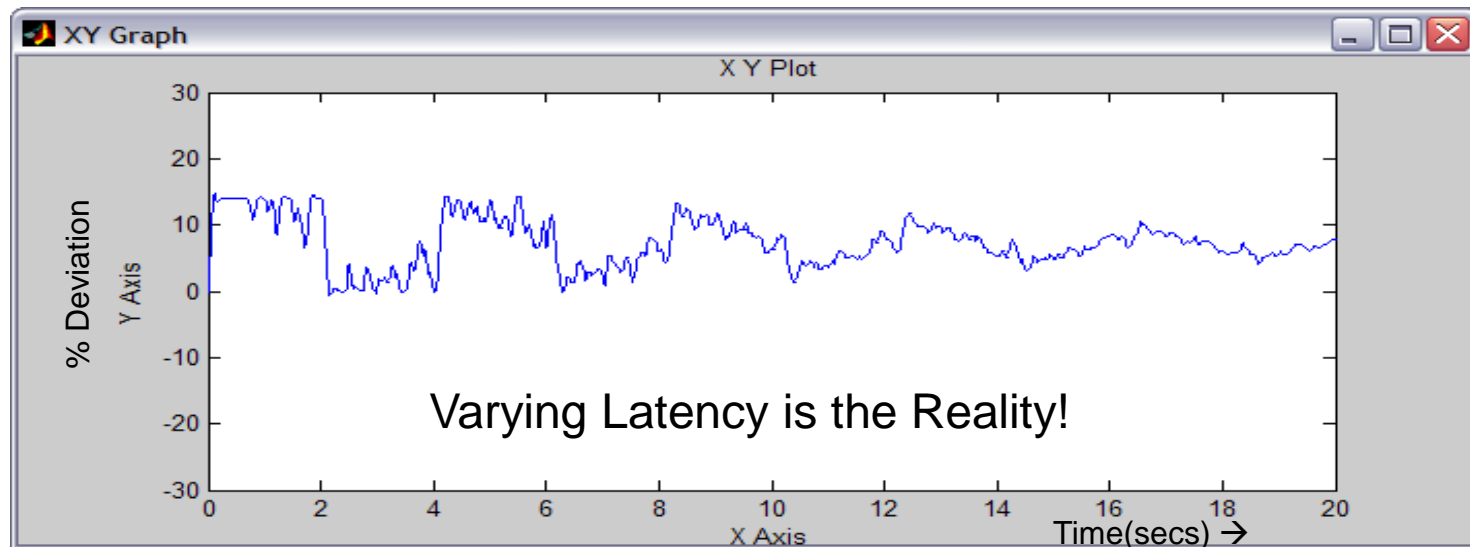
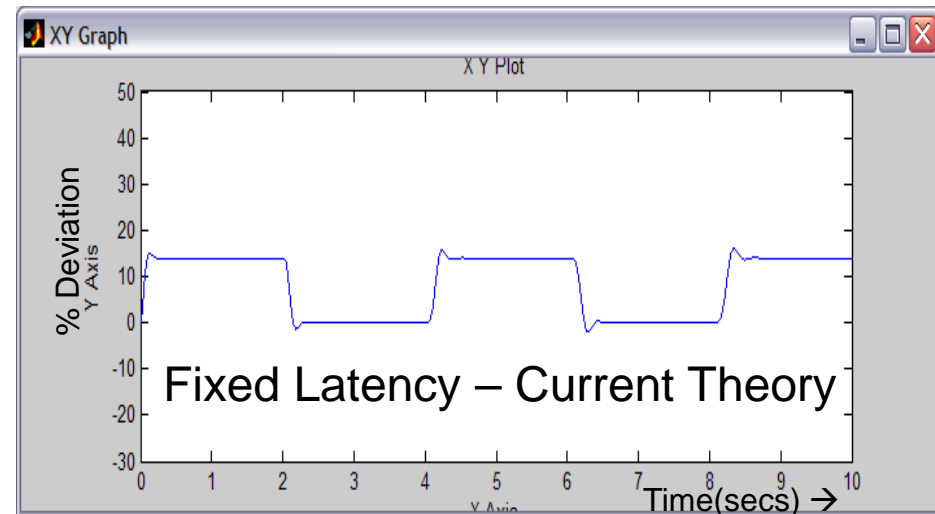
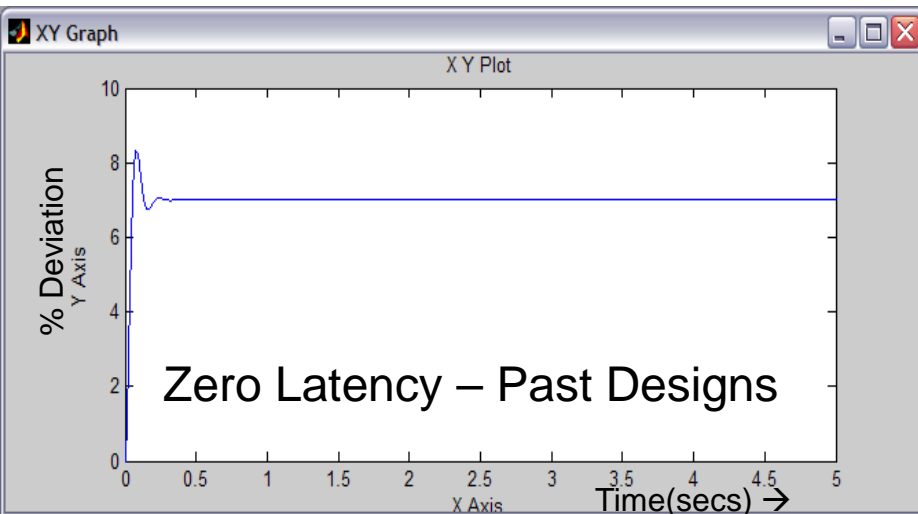


Instrumentation

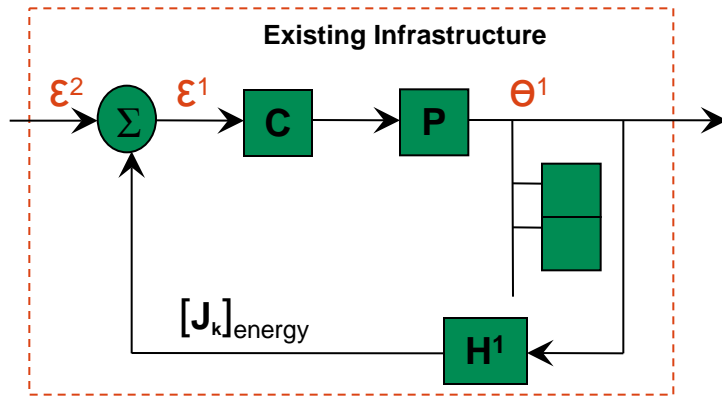
- Refrigeration system is fully instrumented
 - Refrigerant temperature, pressure and flow rate
 - Compressor power
 - Display case power (fans, lighting, defrost heater)
 - Display case discharge and return air temperatures
 - Simulated product temperatures



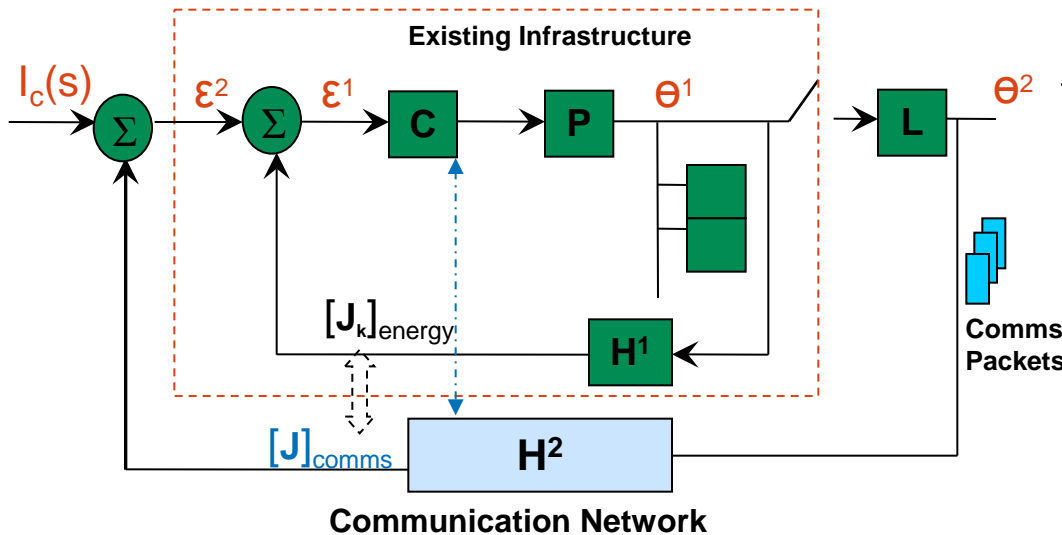
Communications Affects Control Stability



Communications Network Introduces New Terms in Stability Equation



$$\frac{\theta^1(s)}{\varepsilon^2(s)} = \frac{C(s) * P(s)}{1 + H^1(s) * C(s) * P(s)}$$



$$\frac{\theta^2}{I} = \frac{LCP}{1 + H^1CP + \mathbf{LH^2 + H^1H^2LCP}}$$

Influence Terms: **< LH² , H¹H²LCP >**

LH² : Coupling of two highly unstable/non-deterministic elements

H¹H²LCP : Cross-couple of networks