Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

VOLTTRON™ Integration with MATLAB, FNCS, Energy+ & GridLAB-D

POORVA SHARMA

Pacific Northwest National Laboratory

VOLTTRON™ 2016

Goal: Incrementally Increasing Realism

- Proof of Concept

- Simulation Testing

- Real Deployment

From Theory to Deployment: The Tools

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

- FNCS
  - A framework for integrating simulators across multiple domains
  - Enables multi-scale modeling & simulation
  - Allows for re-use of existing software AND models

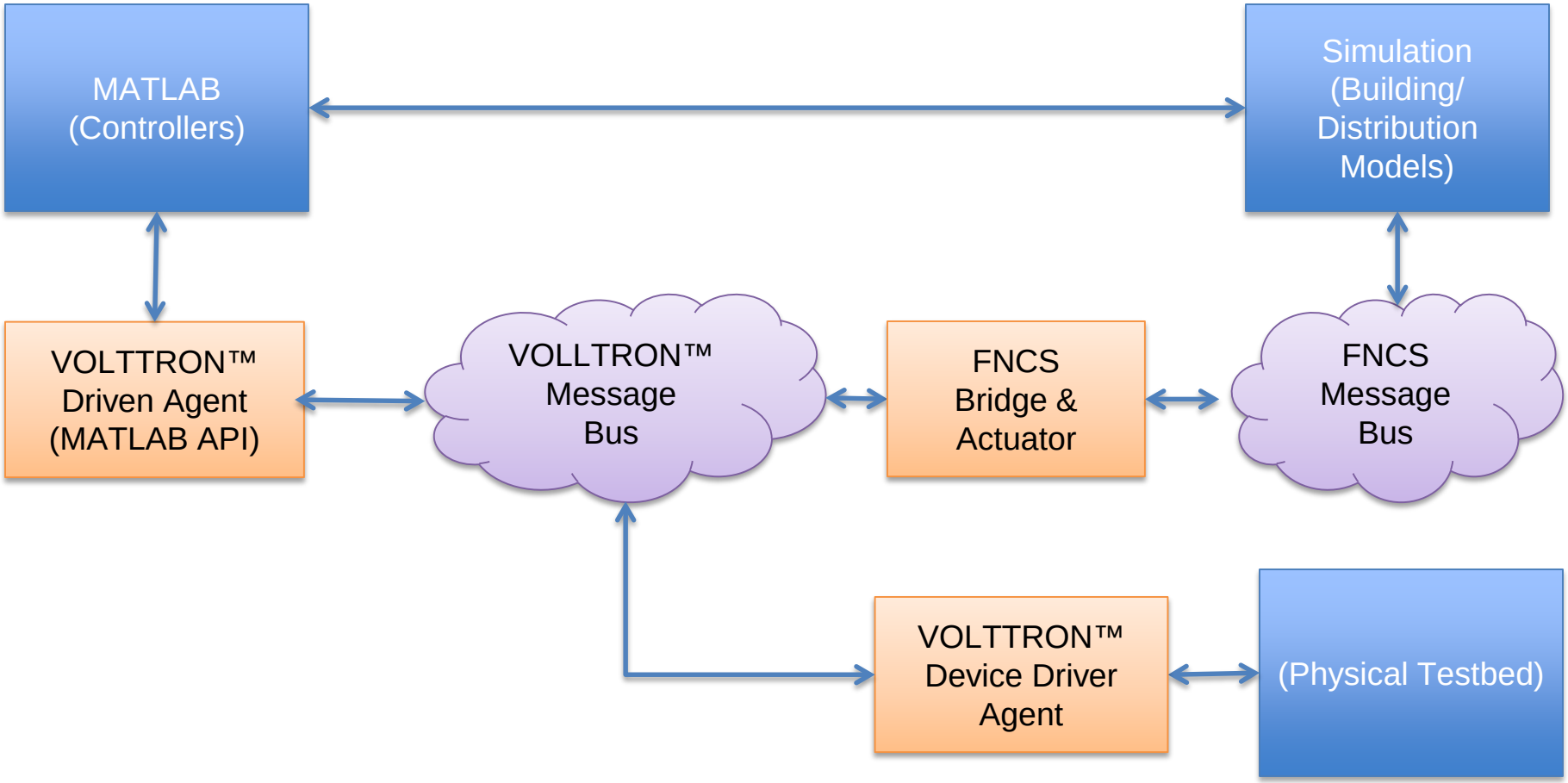
- Energy+
  - A whole building energy simulation program
  - Supports big and complicated building models design.

EnergyPlus

- GridLAB-D
  - Distribution system simulation and analysis tool
  - Incorporates advanced modeling techniques

GridLAB-D

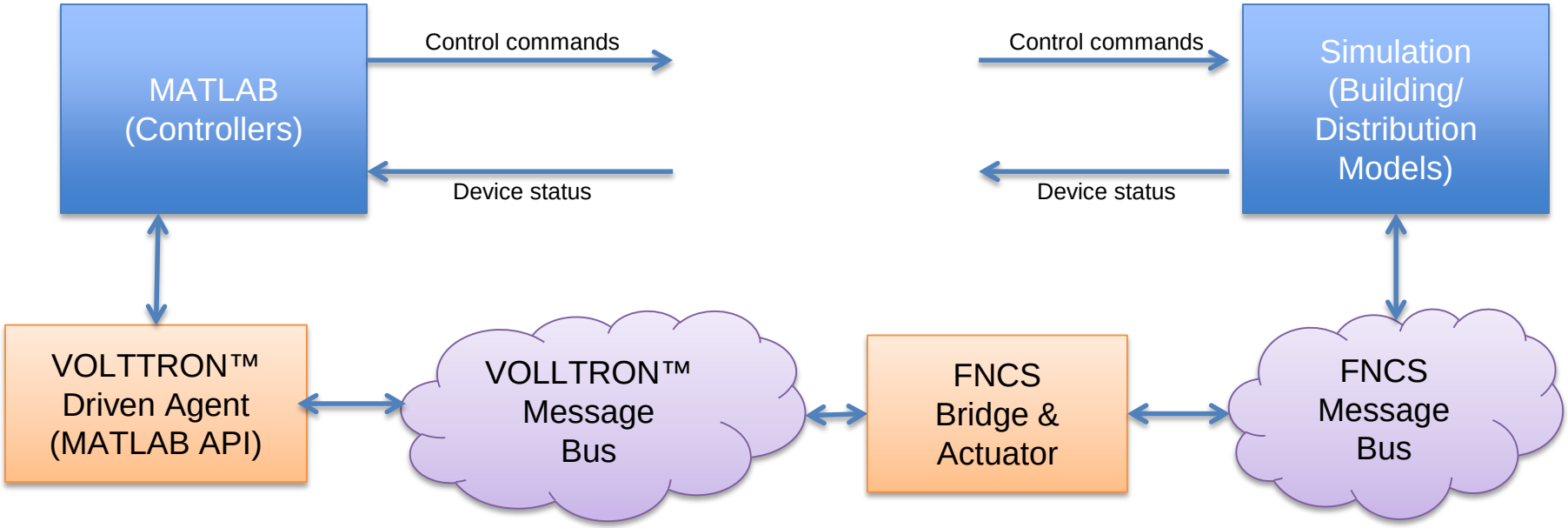
Intentional Experimental Design



MATLAB-VOLTTRON™ Connection

- Data Flow
  - Devices publish their data after specified time on VOLTTRON™ message bus.
  - Driven Agent subscribes to devices mentioned in config file and forwards their data to MATLAB application.
  - MATLAB application receives device data messages, do the computation and sends set point commands to Driven Agent.
  - Driven Agent publishes set point commands to VOLTTRON™ message bus.
  - These commands are received by device drivers to set point on device.

VOLTTRON™-FNCS Connection

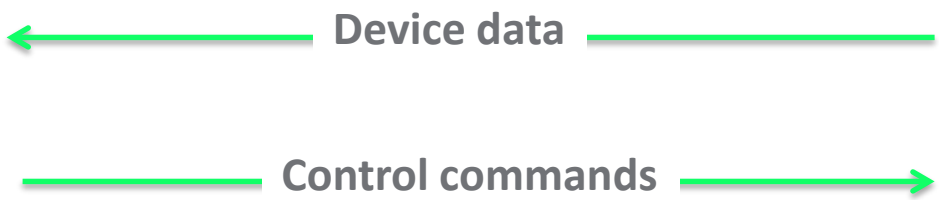


- VOLTTRON™ FNCS Bridge is a FNCS client and a VOLTTRON™ agent
- Bridge is accelerated – one second tick equates to 5 minute model time
  - Want to test control code as fast as possible
  - Later deploy in real-time

Demo Run

- Left side - MATLAB
  - Control application
- Output
  - Receiving device data
  - Sending control commands

- Right side – Run Script:
  - VOLTTRON™
  - FNCS
  - Energy+
  - VOLTTRON™ Driven Agent (MATLAB API)
  - VOLTTRON™ FNCS Agent
- Output
  - Sending device data
  - Receiving control commands



VOLTTRON™ – MATLAB Interface File Structure

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

MATLAB

- example.m

DrivenMatlab Agent

- DrivenMatlabAgent
  - drivenmatlabagent
    - drivenagent.py
    - matlab.py
  - config_file
  - setup.py

Note: MATLAB controller and VOLTTRON™ running on different machines.

VOLTTRON™ – MATLAB Interface Setup/Configuration

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

MATLAB

- Setup Python in MATLAB
  - Install Python
  - Install pyzmq
  - Setup Python path in MATLAB

    pyversion [path/to]/python.exe
- Get example.m

DrivenMatlab Agent

- Create configuration file

```
{  "agentid": "matlab",
   "application": "drivenmatlab.matlab.Application",
   "device": {
      "campus": “CampusName",
      "building": “BuildingName",
      "unit": { "HEATER1": {  "subdevices": [] },
                "HEATER2": {  "subdevices": [] }
      },
      "analysis_name": "WaterHeater_Controller"
   },
   "mode": "ACTIVE",
   "arguments": {
         "status_stpt": "statussetpoint",
         "temperature": "temperature",
         "config_url": "tcp://[ip address]:5556",
         "data_url": "tcp://[ip address]:5557",
         "recv_timeout": 50000
   },
   "conversion_map": {
      "statussetpoint*": "bool",
      "temperature*": "float"
   },
   "unittype_map": {
      "statussetpoint*": "On/Off",
      "temperature*": "Farenheit"
   }
}
```

VOLTTRON™ – MATLAB Interface
Steps to run

- Start VOLTTRON™ platform

  ./env/bin/activate
  volttron –vv

- Start Actuator agent

  export SOURCE=services/core/ActuatorAgent
  export CONFIG=services/core/ActuatorAgent/actuator-deploy.service
  export TAG=actuator
  ./scripts/core/make-agent.sh

- Start MasterDriver Agent

  export SOURCE=services/core/MasterDriverAgent
  export CONFIG=services/core/MasterDriverAgent/fake-master-driver.agent
  export TAG=masterdriver
  ./scripts/core/make-agent.sh

- Start DrivenMatlabAgent

  export SOURCE=applications/pnnl/DrivenMatlabAgent
  export CONFIG=applications/pnnl/DrivenMatlabAgent/config_waterheater
  export TAG=matlab
  ./scripts/core/make-agent.sh

- Run MATLAB example controller

  example.m

VOLTTRON™ – MATLAB Interface Data Flow

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

MATLAB

- Setup sockets

```
config_url = 'tcp://*:5556';
data_url = 'tcp://*:5557';
```

- Send configuration request

```
message = 'config';
config_socket.send_string(message,py.zmq.NOBLOCK);
```

- Receive configuration

```
config_params = config_socket.recv_json();
disp(config_params)
```

DrivenMatlabAgent

- Setup socket (matlab.py)

```
self.config_socket = context.socket(zmq.PAIR)
self.config_socket.connect(config_url)
self.data_socket = context.socket(zmq.PAIR)
self.data_socket.connect(data_url)
```

- Receive configuration request

```
event = self.config_socket.poll(self.recv_timeout)
```

- Send configuration

```
self.config_socket.send_json(kwargs,zmq.NOBLOCK)
```

VOLTTRON™ – MATLAB Interface
Data Flow (cont.)

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

MATLAB

- Wait for device data

```
event = data_socket.poll(recv_timeout);
```

- Receive device data

```
if event > 0
    disp('Receiving data')
    data = data_socket.recv_pyobj();
```

- Send control command

```
matlab_result = '{"commands":
                 {"Zone1":[["temperature",27]],
                 "Zone2":[["temperature",28]]}}';
disp('sending matlab result')
data_socket.send_json(matlab_result,py.zmq.NOBLOCK)
```

DrivenMatlabAgent

- Subscribe to device data

```
self.vip.pubsub.subscribe(peer='pubsub',
                          prefix=device_topic,
                          callback=self.on_analysis_message)
results = app_instance.run(_timestamp, device_data)
```

- Send device data

```
self.data_socket.send_pyobj(points,zmq.NOBLOCK)
```

- Receive control commands and call Actuator's set point

```
print("Waiting for matlab results")
event = self.data_socket.poll(self.recv_timeout)

self.actuation_vip.call('platform.actuator', 'set_point',
                        actuator_id, point_path,
                        new_value).get(timeout=15)
```

VOLTTRON™ – FNCS Interface

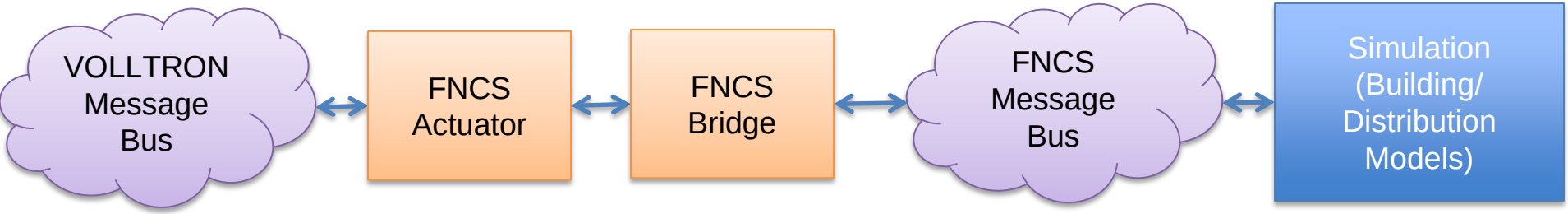
Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

FNCS Actuator Agent

- FncsActuatorProxy.py
- It is an Actuator agent specific to FNCS co simulator tool.

FNCS – Simulation Bridge

- FNCS_Volttron_Bridge.py
- Communication bridge between simulation and Actuator agent



VOLTTRON™ – FNCS Interface Configuration

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

FNCS Actuator Agent

```
FncsProxy(identity='platform.actuator')
```

FNCS – Simulation Bridge

- FNCS_Volttron_Bridge.py

```
{“simulation_run_time" : "1d",
"heartbeat_period" : 10,
"heartbeat_multiplier" : 15,
"fncs_zpl" : {
                "name" : "FNCS_Volttron_Bridge",
                "time_delta" : "150s",
                "broker" : "tcp://localhost:5570",
                "values" : {
                “point_name" : {
                "topic" : “device_name/point_name",
                "default" : "0",
                "type" : "double",
                "list" : "false"
                },…}},
"remote_platform_params" : {
                "vip_address": "tcp://[ipaddress]",
                "port": 55055,
                "agent_public": “[key]",
                "agent_secret": “[key]",
                "server_key": “[key]"          }}
```

VOLTTRON™ – FNCS Interface
Steps to Run

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

- Start VOLTTRON™ platform
  ./env/bin/activate
  volttron –vv
- Start FNCS_Volttron_Bridge & FncsActuatorProxy
  python FNCS_Volttron_Bridge.py
  python FncsActuatorProxy.py
- Run Simulation
  Energy+ or Gridlab-D
- Start FNCS broker
  fncs_broker 2
- Start DrivenMatlabAgent
  export SOURCE=applications/pnnl/DrivenMatlabAgent
  export CONFIG=applications/pnnl/DrivenMatlabAgent/config_waterheater
  export TAG=matlab
  ./scripts/core/make-agent.sh
- Run MATLAB example controller
  example.m

VOLTTRON™ – FNCS Interface Data Flow

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

FNCS Actuator Agent

- Receive device data, publish on VOLTTRON bus

```
def onmessage(self, peer, sender, bus, topic, headers, message):
    # listen to fncs/output/devices
    # publish to devices
```

- Receive control commands, publish on FNCS bus

```
@RPC.export
def set_point(self, requester_id, topic, value, **kwargs):
    # publishes to the fncs/input/ subtopic for information
    # that goes to the bridge to pass to fncs message bus
```

FNCS – Simulation Bridge

- Publish device data

```
def publish_heartbeat(self):
    '''Send heartbeat message every HEARTBEAT_PERIOD seconds.
```

- Receive control command forward to simulation

```
def onmessage(self, peer, sender, bus, topic, headers, message):
    d = {'topic': topic, 'headers': headers, 'message': message}
    # Forward message to FNCS
```