# Scalability Application

The purpose of the scalability experiment is to demonstrate the use of Volttron in a distributed environment. The setup would consist of many Volttron nodes communicating with each other to achieve a control objective. The demonstration application chosen is a distributed signal regulation algorithm. There are two distinct components to the scalability experiment: *(i****)*** Distributed implementation of a control algorithm regulation that meets a regulation signal and *(ii)* Leader selection algorithm to provide robustness. The section below explains the control algorithm in detail.

## Distributed signal regulation agent

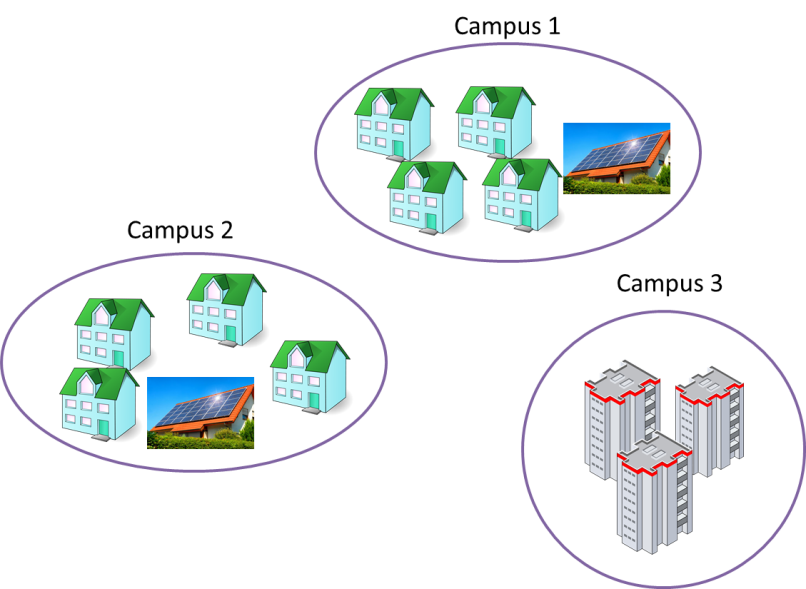
Buildings are currently large passive participants in the electric grid. They comprise a large thermal storage that can be used as an ancillary service to enhance the reliability of the power grid, which will be subject to more volatility and unpredictability from the integration of renewable energy resources. Two control strategies have been explored which allow HVAC loads in commercial and residential buildings to provide frequency regulation services to the grid while maintaining occupants’ comfort. The first control strategy is based on model predictive control acting on a variable air volume HVAC system which is available in large commercial buildings. The second strategy is a rule-based control which acts on many HVAC units switching them on or off. These units are typically available in residential buildings and in many small to medium size commercial buildings.

Recently, the simulation experiments were implemented in Matlab and considered a set of 50 buildings which illustrated the proposed control coordination and control of a large number of HVAC systems for commercial and residential buildings. In this setup, a master control would send on/off signals to control HVAC units to meet a signal regulation objective. Ordinary differential equations (ODEs) were used to model the HVAC units and the master control would determine the number of units that must be switched on or off. The individual units were selected in a manner to minimize occupant discomfort.

The developed algorithm was ported to Volttron and implanted with additional features to enable true multi-node control coordination. The implementation consisted of a MasterNode Agent and multiple ModelNode Agent instances. Upon booting up, a ModelNode instance would communicate with the MasterNode periodically updating the master of its temperature and HVAC state. The MasterNode maintains a list of active ModelNodes, aka, a list of buildings being controlled. The first time a ModelNode communicates with the MasterNode is considered a new ‘registration’ and the ModelNode gets added to the list of ModelNodes to be controlled. The MasterNode senses the regulation need every x minutes (can be 10 minutes to avoid short cycling equipment) and sends out control signals to the ModelNodes.

The developed algorithm was tested on multiple instances of Volttron running on Oracle VirtualBox[[1]](#footnote-1) as well as more extensively using the ADEVS (A Discrete Event Simulation Framework) that is tied to a buildings model developed in Modelica. While this demonstrated the scalability of typical algorithms in Volttron, true scalability should be demonstrated through the implementation of two principles: *(i)*Election of a leader for each model node *(ii)*Achieving robustness.

In a typical real-world control implementation, there is a high likelihood of some form of master node in a building/campus. Grid level needs should be met robustly across an integrated set of buildings, microgrids. Therefore, it is important to design distributed transactive algorithms that allows for flexibility within a customer’s premises as well as integrates seamlessly across multiple customers. The following diagram illustrates the scenario.



*Fig: A campus may consist of multiple buildings, each controlled by a Volttron node in the same network. Each campus itself elects a master node.*

Therefore, within a campus, these are the steps such an algorithm should accomplish:

1. Establish a virtual network, could use the existing wired or wireless network. In a simple implementation, a wireless ad-hoc network would suffice.
2. Individual Volttron nodes (possible master nodes and model nodes), upon boot up, will register in the network.
3. Masters send heartbeat messages to all model nodes.
4. Nodes determine (elect) a leader among possible masters determined from heartbeat messages. A bully algorithm[[2]](#footnote-2) is implemented that uses the node that has the smallest/largest IP that is up and running.
5. Each node determines their master.
6. Each master node knows the regulation capacity/flexibility for the network it resides in.
7. Nodes communicate periodically with the leader (sending temperature values).
8. Masters run optimization algorithms and communicate control decisions back to model nodes.
9. Models leverage the control decisions from the determined leader only.

Our implementation focuses on a single campus and includes multiple presumptions. First, all model and master node IPs are determined and model nodes know possible master node IPs in advance (in the config file) so that they can register and send temperature values periodically. This is found to be a Volttron requirement. Second, master nodes are given the regulation signal in advance for the network/building/campus they are giving decisions for. Therefore, master nodes give the same decisions based on the same regulation signal and received temperatures from the model nodes. Based on masters’ leader selection heartbeats, models determine a master that is up and running. Basically, robustness is realized from redundancy. Third, each campus is given the corresponding regulation signal files in advance and masters can read that file. One possible extension to this implementation would be to implement an upper level master node that automatically distributes the grid-level regulation signal into chunks for different campuses or buildings. It can be realized by manual I/O operations for now that allocates the fixed grid regulation signal to each campus. In the following section, deployment instructions of the test case are listed.

## Deployment of the test case

A key technical achievement in facilitating the creation of the testbed was creating a very small footprint system image running Volttron. Significant effort went into its creation, including dynamic repartitioning of a tinycore Linux installation, compiling required compilers in tinycore to create Python modules, as well as modifications to the system’s library loading routines. These efforts resulted in a system image that is much under 200MB and can run in as little as 256 MB RAM, while enabling full networking, serial ports, and additional peripherals for true controls emulation.

For verification, we installed the model agent to two different tinycore images and installed master agent to two another tinycore images. We ran all four agents in VirtualBox registering to the same internal network[[3]](#footnote-3) and assigned their IPs in the startup script (.profile script). Please note that model nodes must include the master IPs in their config file. Additionally, each assigned IP should be written to the platform’s config file (/<INSTALLATION PATH>/.volttron/config). Necessary code snippets are included in the startup script in the image files. When the images are up and running, the Volttron platforms and the installed agent are started consecutively. We tested the communication of decisions and leader assertion results. Additionally, we took the leader node (a master) down and observed the leader assignment changes to the other master node achieving uninterrupted decision flow for the HVAC system.

# Appendix

The agent files and the tinycore images are shared in the code repository (<https://code.ornl.gov/Sim_based_testing/scalability_test.git>). The repo is public so it could be cloned without authentication. However, if subscription to the repo is not successful or there is a need for pushing some changes, please go to xcams.ornl.gov and create a username and a password. Contact [ozmeno@ornl.gov](mailto:ozmeno@ornl.gov) or [sanyalj@ornl.gov](mailto:sanyalj@ornl.gov) to gain access to the repository.

If you would like to replicate the test-case, please clone replicates of the shared images and run multiple of them on the same internal network after making the necessary IP and config file changes as stated above in the Sect 1.2.

1. https://www.virtualbox.org [↑](#footnote-ref-1)
2. https://en.wikipedia.org/wiki/Bully\_algorithm [↑](#footnote-ref-2)
3. Other than doing it manually, one way is to use vboxmanage dhcpserver command to set-up an internal network with lower and upper IPs. Images will get their IPs in an order when they are up and running. This way user/developer would be able to control which IPs master nodes and what model nodes get. Another solution is to use MAC addresses to assign IPs. [↑](#footnote-ref-3)