CLEAVE Paper for CNERT'21

Manuel Olguín Muñoz
School of Electrical Engineering
and Computer Science
KTH Royal Institute of Technology
Stockholm, Sweden

James Gross
School of Electrical Engineering
and Computer Science
KTH Royal Institute of Technology
Stockholm, Sweden

***Abstract*—**

TODO

I. Introduction

II. The CLEAVE Framework

CLEAVE (*ControL bEnchmArking serVice on the Edge*) is our framework for quick, robust, and repeatable benchmarking of networked control systems. CLEAVE tackles this challenge through a number of key design decisions that we will discuss in the following.

Emulation approach. Existing benchmarking platforms for NCS either take a fully simulated approach — running plant, network, and controller inside a completely simulated environment — or one requiring an actual physical plant that interacts with a real network and controller. CLEAVE walks the line between these two approaches, employing a strategy based on having a real-time emulated representation of a Plant interact with the real network and controller.

Plant- and Controller-agnosticism. CLEAVE makes as few assumptions as possible about the inner workings of both the Plant and the Controller. The only requirements for a system to be able to be emulated in CLEAVE are 1) that its behavior can be described in a discrete-time fashion; and 2) that its state and the actions performed on it can be described by a finite number of scalar and/or vector variables.

API based on well-defined components. The previously mentioned agnosticism is achieved through careful abstraction of the the system into an API based around the following components:

- A `State` object, which implements the discrete-time evolution of the physical system being controlled. This object also holds a number of named properties that can be measured or actuated upon.
- An arbitrary number of `Sensor` objects which measure named properties of the `State`, and potentially transform them (for instance, to add noise).
- A `Controller` object which receives values of named properties processed by the `Sensor` objects and returns new values for the named properties of the `State` that can be actuated upon.
- Finally, an arbitrary number of `Actuator` objects which receive the values generated by the `Controller` and process them before passing them on the `State`.

Users implement their desired behavior by extending the abstract base classes defining these components provided by the framework.

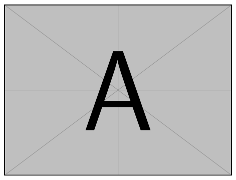
Fig. 1: Execution flow of a time step in CLEAVE.

```python
from cleave.api.plant import
    SimpleConstantActuator, SimpleSensor
from cleave.impl.inverted_pendulum import
    InvPendulumState

host = 'localhost'
port = 8080

controller_class = 'InvPendulumController'

tick_rate = 100
emu_duration = '30m'

state = InvPendulumState(fail_angle_rad=-1)

sensors = [
    SimpleSensor('position', 100),
    SimpleSensor('speed', 100),
    SimpleSensor('angle', 100),
    SimpleSensor('ang_vel', 100),
]

actuators = [
    SimpleConstantActuator(initial_value=0,
        prop_name='force')
]

output_dir = './plant_metrics'
```

Listing 1: Example configuration file.

Plain Python configuration. As mentioned above, users implement their desired behavior for the core components by extending classes provided by the framework. This is done in configuration files written in full-featured Python simply defining a number of required top-level variables and which are then passed to the main script provided with CLEAVE. This allows for easy configuration of arbitrarily complex emulations.

Fill in core concepts

Link to repo

A. General architecture

The general architecture of CLEAVE centers around two components:

1) the Plant, corresponding to the emulated physical system running on some arbitrary client device;
2) the Controller Service running on some arbitrary cloudlet, which receives samples of the state of the Plant and produces control commands;

These components are connected through the real network

In the following, we will briefly discuss the design and implementation of these components.

1) Plant: As

2) Controller Service:

B. Network

III. Experimental validation

Acknowledgements