Appendix: Artifact Description

1 OVERVIEW OF CONTRIBUTIONS AND ARTIFACTS

1.1 Paper's Main Contributions

C_1 Designing an end-to-end system for coupling sensor networks with HPC facilities through Private 5G networks to enable real-time-digital agriculture simulation.

C_2 Demonstrating the ability to leverage 5G/6G connectivity in remote, low infrastructure settings such as commercial digital agriculture.

C_3 Demonstrating the ability to use a single software stack on all devices in an IoT deployment, at all device scales, including HPC machines.

1.2 Computational Artifacts

A_1 Full Artifact Repository (DOI: 10.5281/zenodo.16696837)

A_2 Full Artifact Repository (github.com/UNL-CPN-Lab)

Artifact ID	Contributions Supported	Related Paper Elements
A_1	C_1, C_3	Figures 3, 7
A_2	C_1, C_2	Figures 4, 5, 6

1.3 Computational Artifact A_1

Relation To Contributions

The artifact provided, (A_1), includes the source code for the simulations to ensure and verify the reproducibility of the artifact. It also includes the plotting tools used to generate the original figures in the paper. The experiments were designed to be run on the head node and a compute node of an HPC cluster.

Expected Results

Figure 3:

The replication of Figure 3 produces a PNG file depicting the final result from the CFD simulation of the CUPS farm. The figure shows the simulation of the airflow around the farm, with the wind velocity represented by color gradients. The simulation is based on a 3D model of the farm.

Figure 7:

The replication of Figure 7 produces a plot depicting the mean total runtime of the OpenFOAM simulation with varying number of cores on a single compute node. The plot shows the mean total runtime for each total number of threads, with error bars representing the 2 standard deviations across multiple runs.

Expected Runtime

The artifact contains the source code and data to reproduce the speedup curve results presented in the paper.

1) *Artifact Setup:* Once downloaded and configured, the artifact can be executed in less than 10 minutes.
2) *Artifact Execution:* The execution time of the artifact can varying greatly depending on queue times, number of jobs submitted concurrently, and number of cores. In total, the execution of one run for Figure 3 took around 15 minutes. For Figure 7, the total execution time was around 13 hours with no queuing delay.
3) *Artifact Analysis:* For Figure 3, there is no analysis to be done. For Figure 7, the analysis of the results and generation of the figures can be done in less than 5 minutes once all the experiments have been executed.

Artifact Setup

1) *Hardware:* Computation is executed on both head nodes and compute nodes. The compute node should have UGE as its batch scheduler. A front-end node with display environment variables is required for rendering the CFD simulations.
2) *Software:* The artifact should be executed on a Linux based machine with Bash, Python, and Pip installed.
3) *Data:* The data for the artifact is provided in a zip file in the directory where it is used. The data includes the all necessary OpenFOAM files.
4) *Installation and Deployment:* The first thing that the user needs to do is to access the head node of an HPC cluster with their display environment variables passed through via SSH. This is accomplished by adding the "-Y" flag when connecting to the front-end (e.g., ssh -Y user@HPC). Next, the user will install the necessary Pip packages listed in the requirements file. The user will run the experiments by running either **"sh runme.sh -t=<number of threads>"** to run a single experiment or run them in batches with **"sh simulations.sh"**. If the user chooses the use the simulations file, they will first have to customize it with how many and which kind of runs they want. Once each experiments are complete, a user can run **"sh render.sh <name of experiment>"** for Figure 3 and/or **"python graphing.py"** for Figure 7. Notably, values are to be copied manually into the CSV file for replications of Figure 7. The values are to be retrieved from the result_time logs. Each line of the CSV file with include: the experiment number and the total time that that the experiment took to run for the varying number of threads.

Artifact Evaluation

For Figure 3:

The user will copy the name of the folder of the completed run (e.g., cups_structure_25-07-29_14_07_57) and use it in the following command: **"sh render.sh <name of folder>"**. This will generate a PNG file in the figures folder showing the completed CFD simulation of the farm. Note: the CFD output in Figure 3 was from a run with 64 threads.

For Figure 7:

The user will run at least 10 simulations for each number of threads for statistical significance. After all runs have finished and the times have been manually copied into the **"data/data.csv"** file, the user will run the **"graphing.py"** file. This will plot and save the graph as a PDF to the figures folder.

Artifact Analysis

The user needs to copy and paste the times from the log file into the CSV files located in **"data/data.csv"**. The user can then run the plotting file located, **"graphing.py"**, to generate and save the figures.

1.4 Computational Artifact A_2

Relation To Contributions

The artifact (A_2) provided contains the source code used to build and deploy the 4G and the 5G network. It also includes a `data` folder containing all the experimental results, as well as plotting code used to reproduce Figures 4, 5, and 6 from the paper.

Repository Structure:

- `build-4G-network/`: Contains source code used to build and deploy the private 4G network
- `build-5G-network/`: Contains source code used to build and deploy the private 5G network
- `data/`: Contains raw `iperf3` JSON output collected during all experiments, categorized by device, duplexing mode, and bandwidth setting.
- `visualize/visualize.ipynb`: Jupyter notebook used to parse the experimental data and generate the plots shown in Figures 4, 5, and 6.

The repository supports full reproduction of the experiments and figures in the paper. Detailed setup and execution steps are provided in the `README.md`.