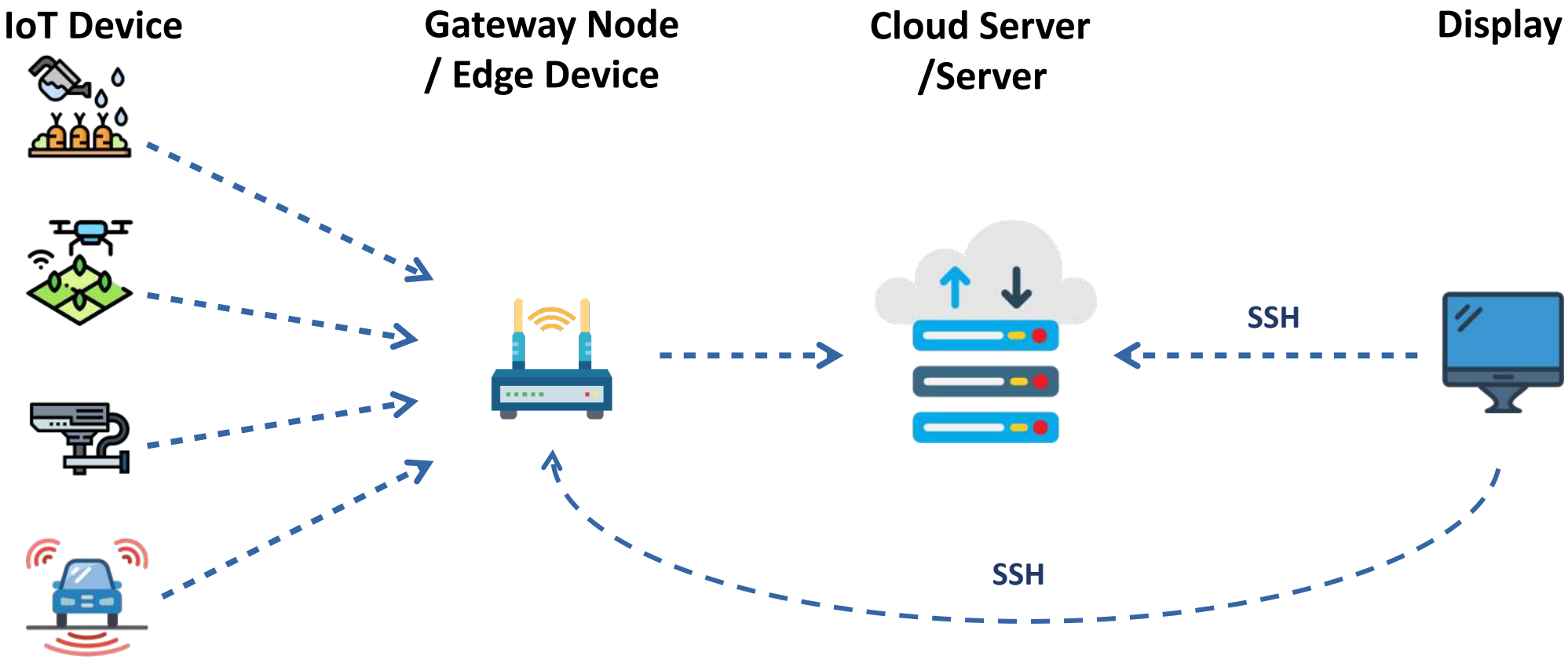
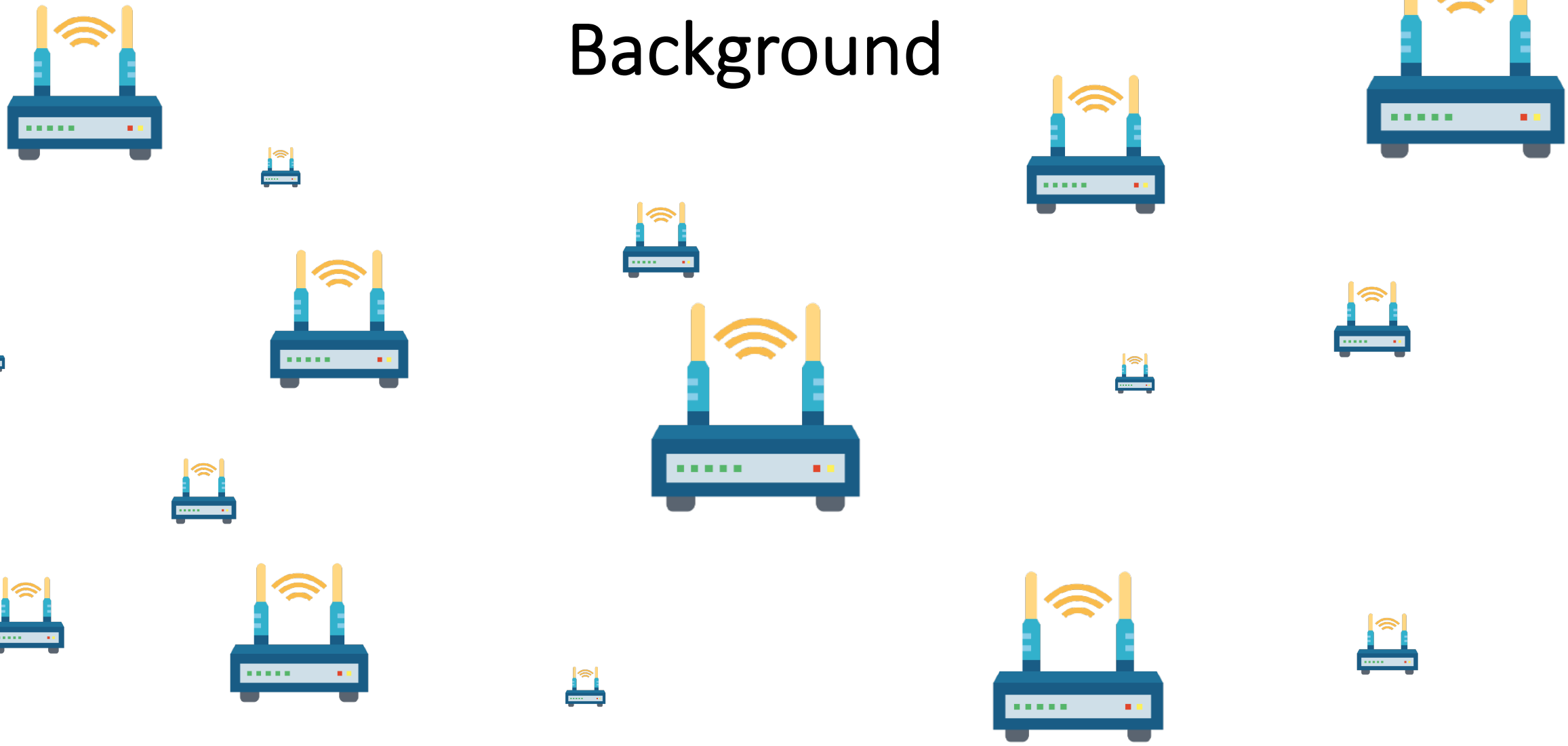
Integrated Application and Performance Monitoring at the IoT Edge

Background


IoT Device
Gateway Node
/ Edge Device
Cloud Server
/Server
Display
SSH
SSH

Background



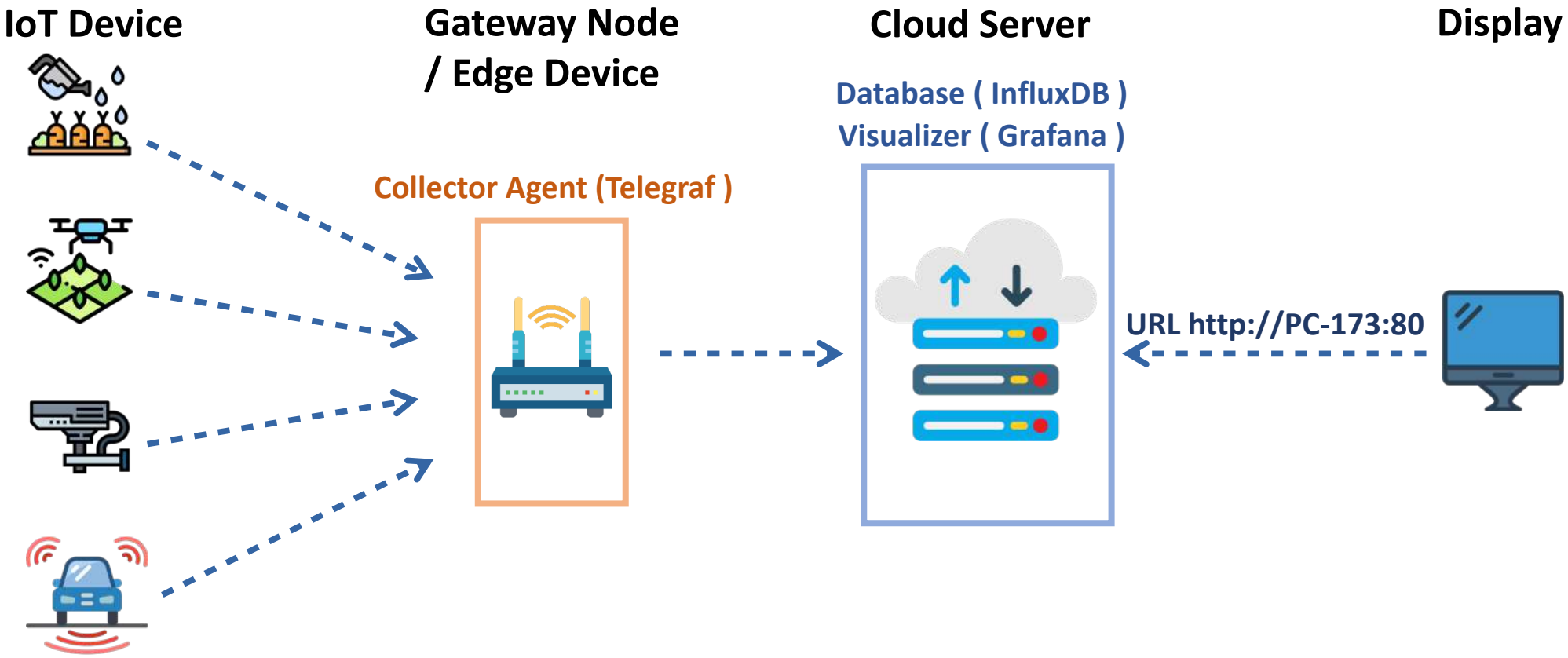
Challenges

- The system must be flexible enough to work with various kinds of commonly used IoT sensors and devices.
- The system must be scalable and power-efficient

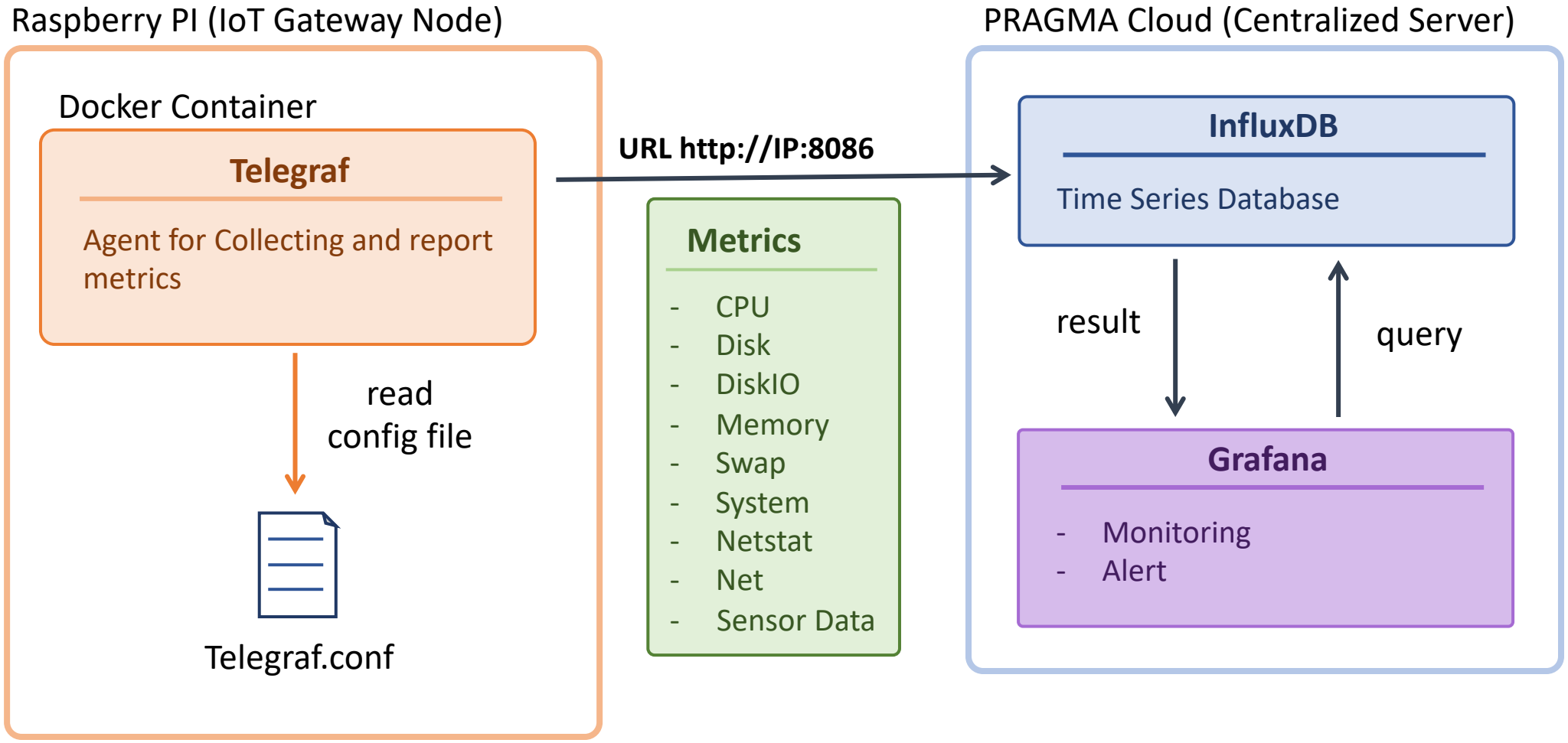
Aims

- Build an integrated solution for IoT infrastructures and applications.

System Architecture

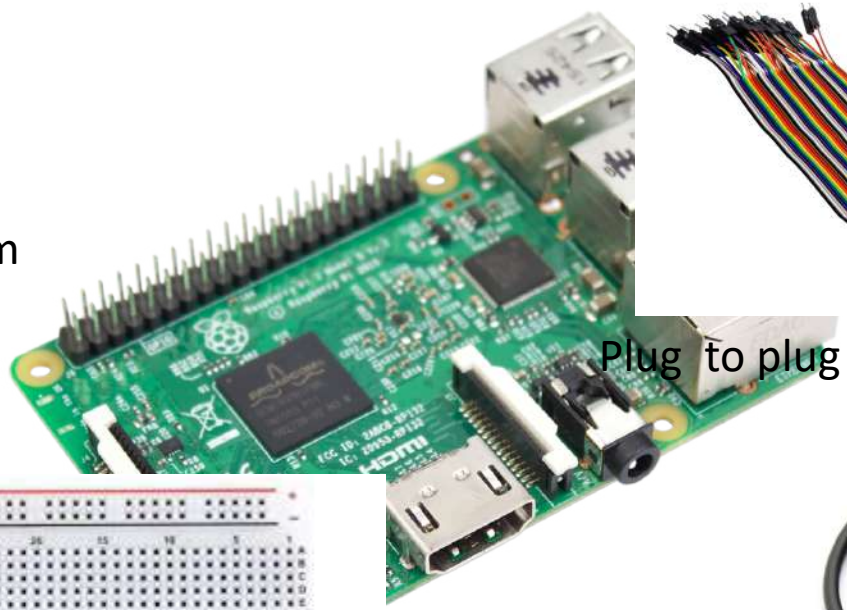


TIG Stack Architecture

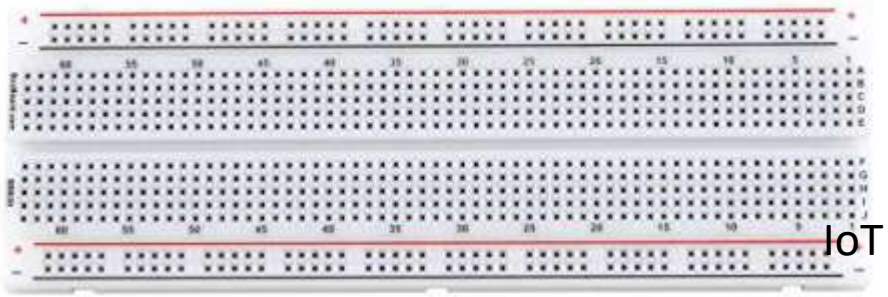


Connecting sensor on Raspberry PI

Resister 4.7k Ohm



Plug to plug cable and Jack to plug cable



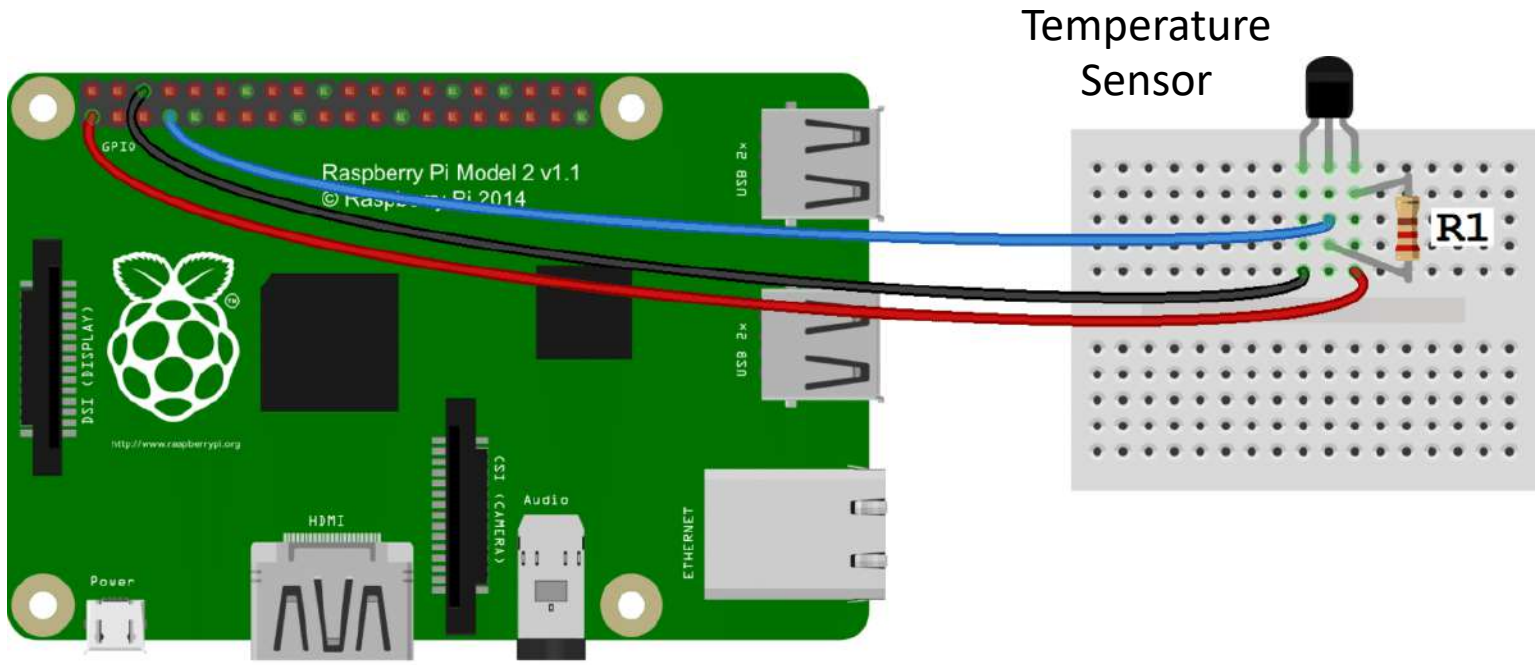
IoT Gateway

Breadboard Circuit

Ds18b20 (Temperature Sensor)

Source: https://www.newteksupply.com/Steren-Arduino-Compatible-15CM-Jumper-Cables-Set--40pcs_p_25251.html

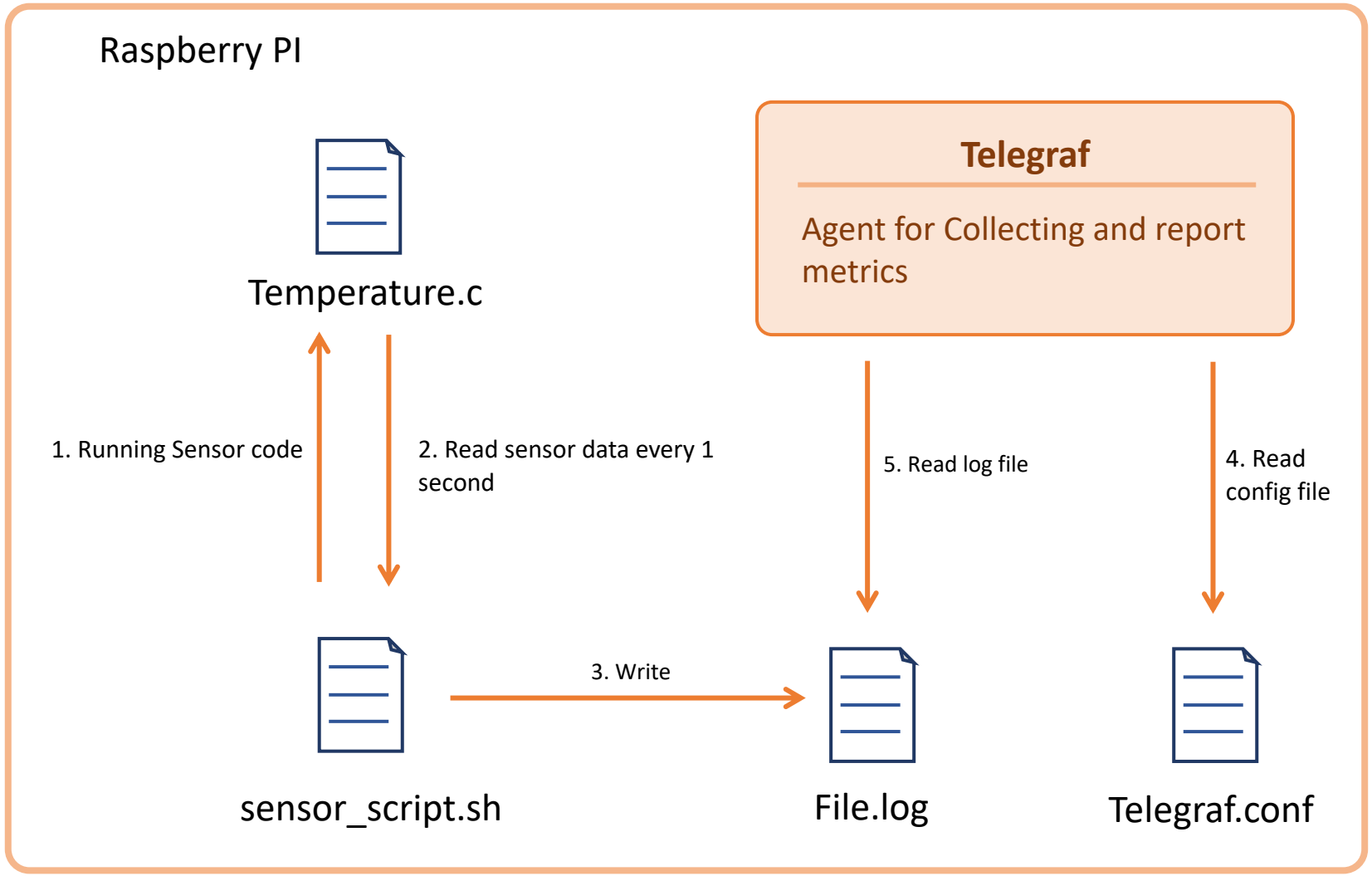
Connecting sensor on Raspberry PI



Circuit Diagram

Source: http://www.circuitbasics.com/raspberry-pi-ds18b20-temperature-sensor-tutorial/

Working flow for Sensor Data Collection

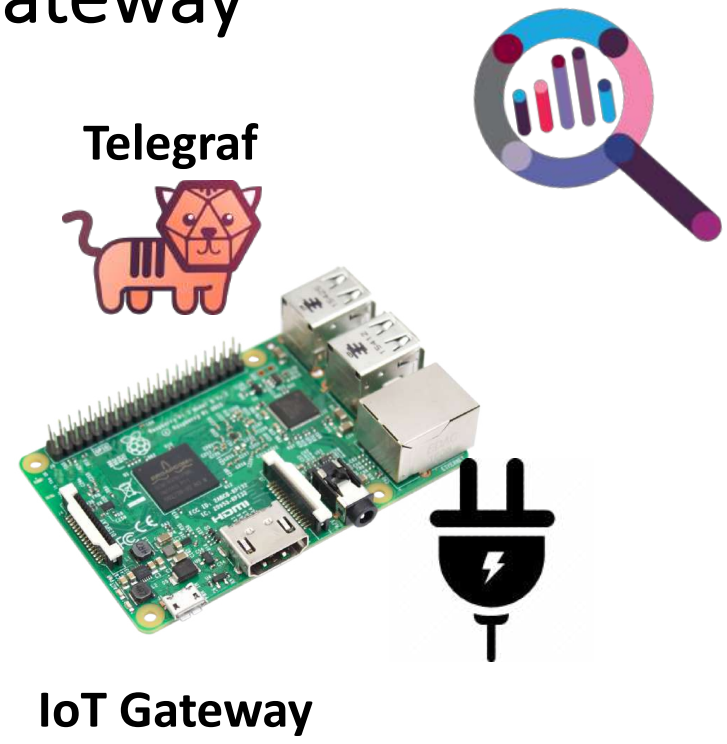


Grafana Dashboard

Purpose of Experiments

Investigate impact of Telegraf in the IoT Gateway

- Measure overhead of Telegraf
- Measure power consumption of IoT Gateway



Resource Utilization and Power Consumption Variables

1. CPU usage (percentage)
2. Memory usage (percentage)
3. Disk I/O usage (percentage)

Resource Utilization

4. Voltage consumption (Volt)
5. Electric current consumption (Ampere)

Power Consumption

Measurement Tools

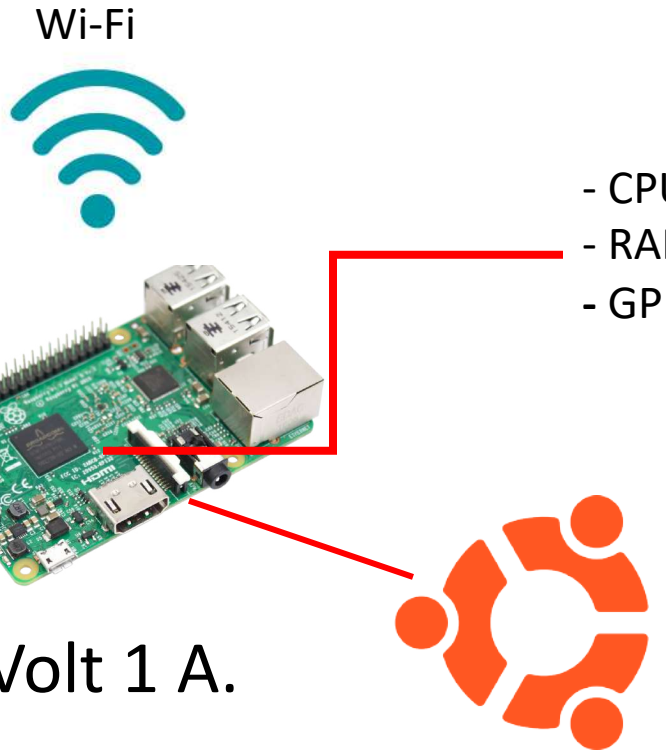
1. sar - Collect, report, or save system activity information.

11:16:25	CPU	%user	%nice	%system	%iowait	%steal	%idle
11:16:26	all	0.25	0.00	3.52	0.00	0.00	96.23
11:16:27	all	0.00	0.00	3.29	0.00	0.00	96.71
11:16:28	all	0.00	0.00	3.29	0.25	0.00	96.46
11:16:29	all	0.25	0.00	3.28	0.00	0.00	96.46
Average:	all	0.27	0.02	3.70	0.00	0.00	96.01

2. USB Power Meter – Collect power consumption such as voltage and current.

Environment

1. Raspberry Pi 3 Model B



2. iPhone Adapter: Output 5 Volt 1 A.

3. Ubuntu Server (Operating System)

Experiment Method

Run each experiment for 15 minutes !!

Repeat each experiment for 5 times !

Reboot Raspberry PI every times when an experiment end !

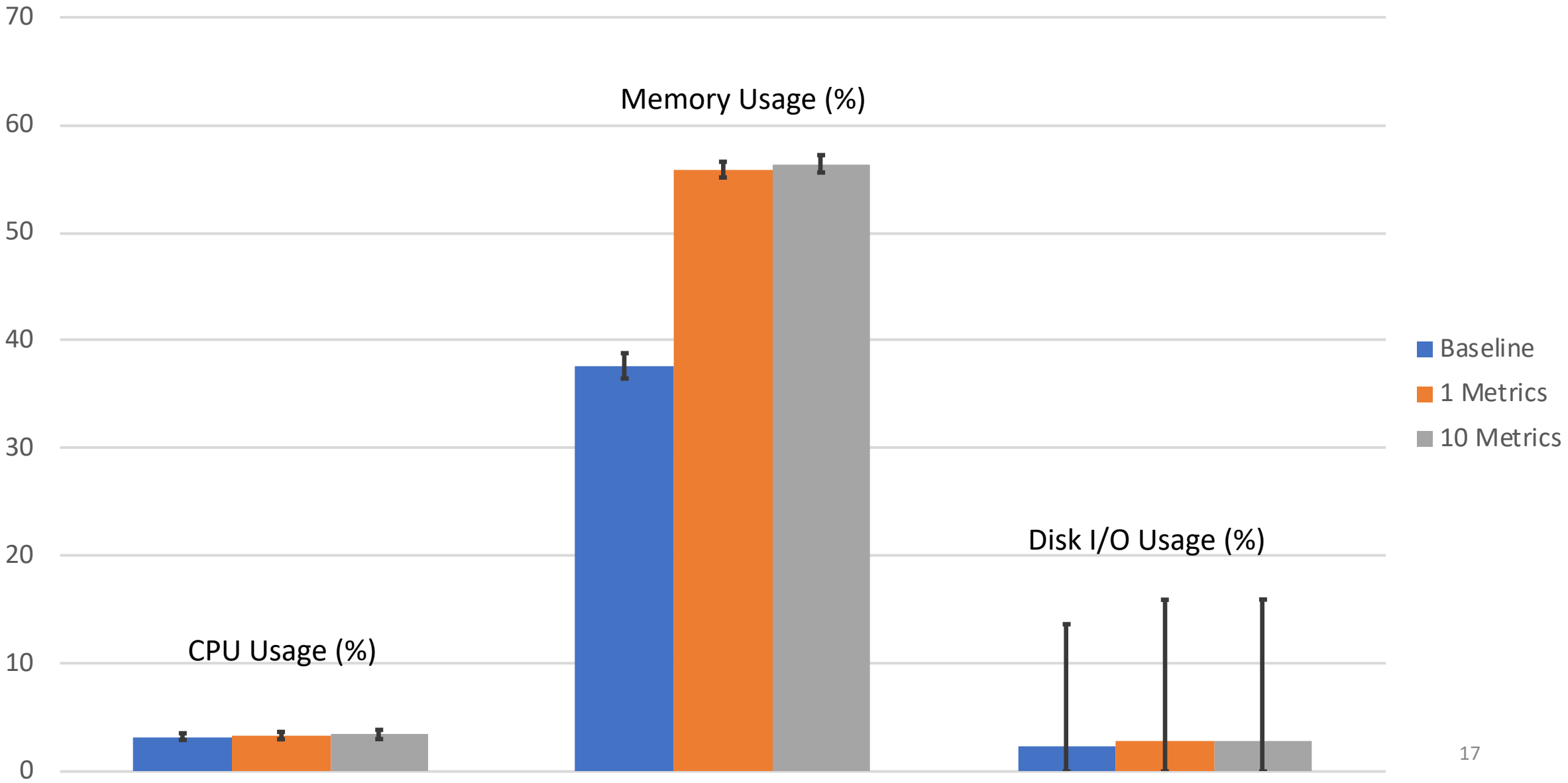
Experiment Setting

Metric = CPU

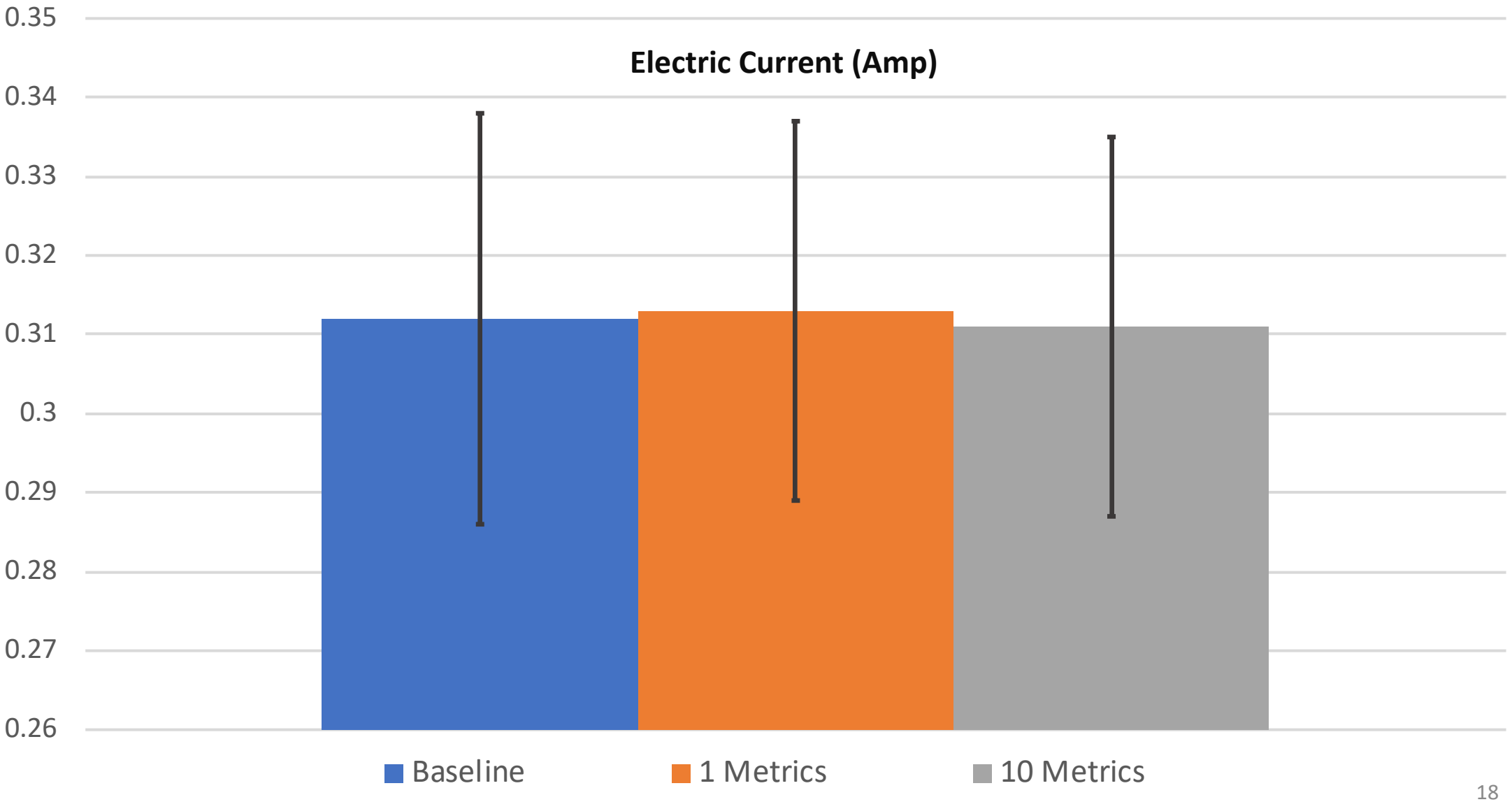
Experiment NO.	Telegraf	Metric	Interval
1.1	-	-	-
1.2	Is Running	1	10 sec
1.3	Is Running	10	

Metric = CPU DISK DISKIO INTERRUPT KERNEL KERNEL_VMSTAT MEM PROCESSES SYSTEM SWAP

Telegraf Overhead (Different Number of Metrics)

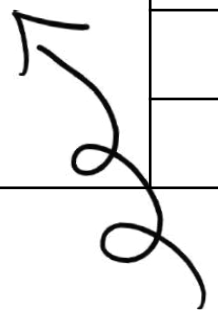


Telegraf Overhead (Different Number of Metrics)



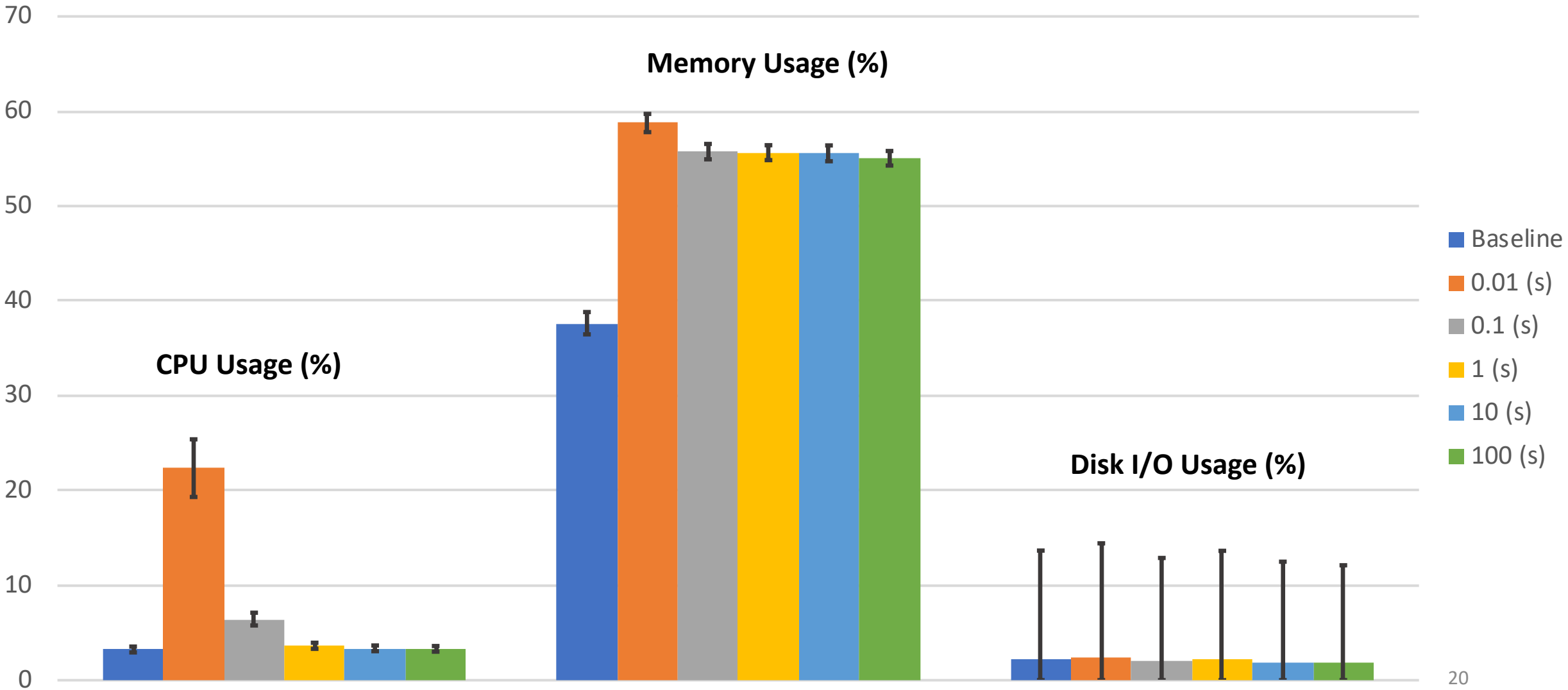
Experiment Setting

Experiment NO.	Telegraf	Metric	Interval
2.1	-	-	-
2.2	Is Running	5	0.01 sec
2.3	Is Running		0.1 sec
2.4	Is Running		1 sec
2.5	Is Running		10 sec
2.6	Is Running		100 sec

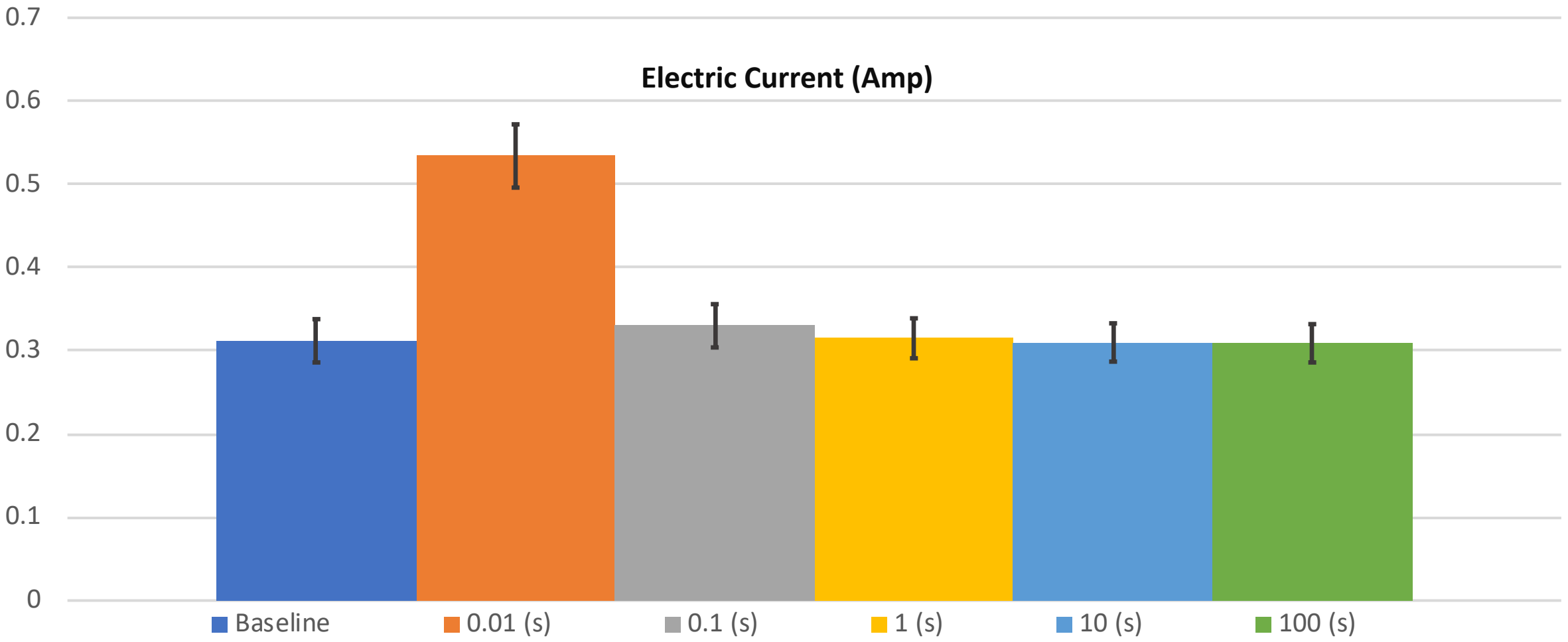


Metric = CPU MEM NET DISKIO SYSTEM

Telegraf Overhead (Different Intervals)



Telegraf Overhead (Different Intervals)

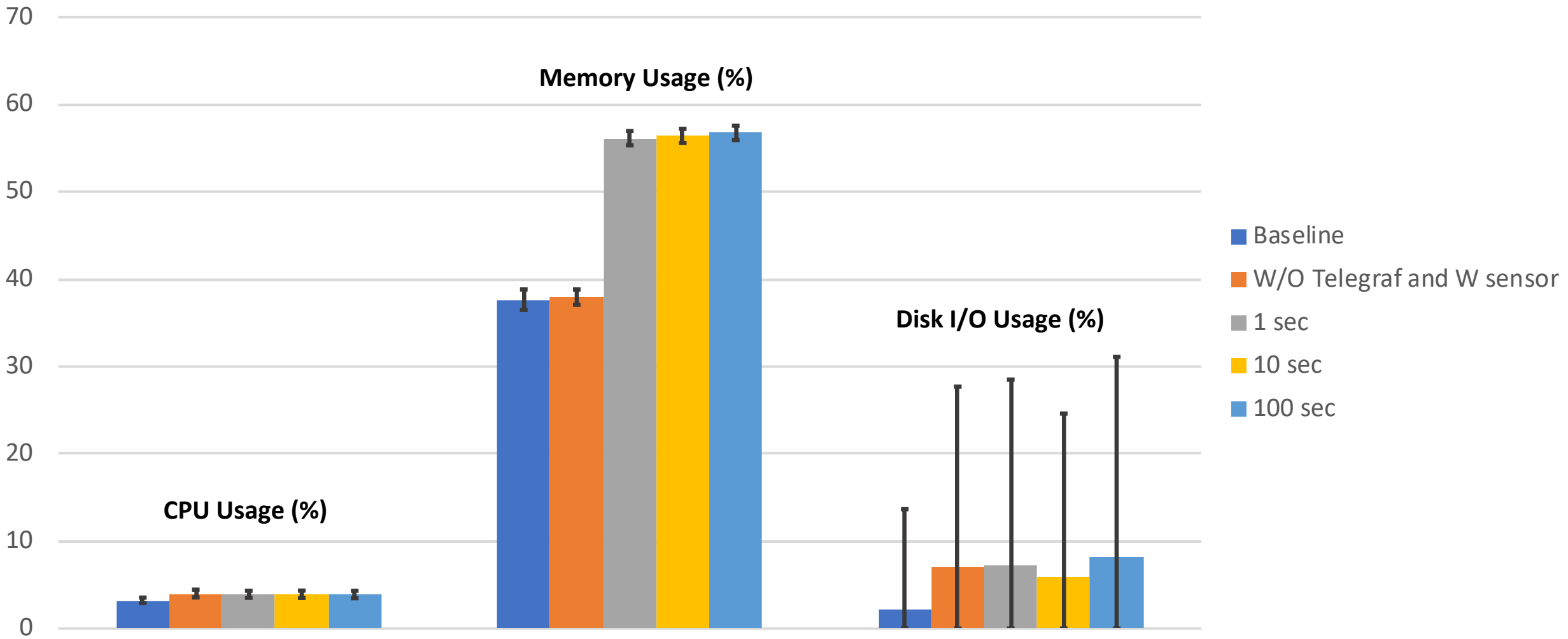


Experiment Setting

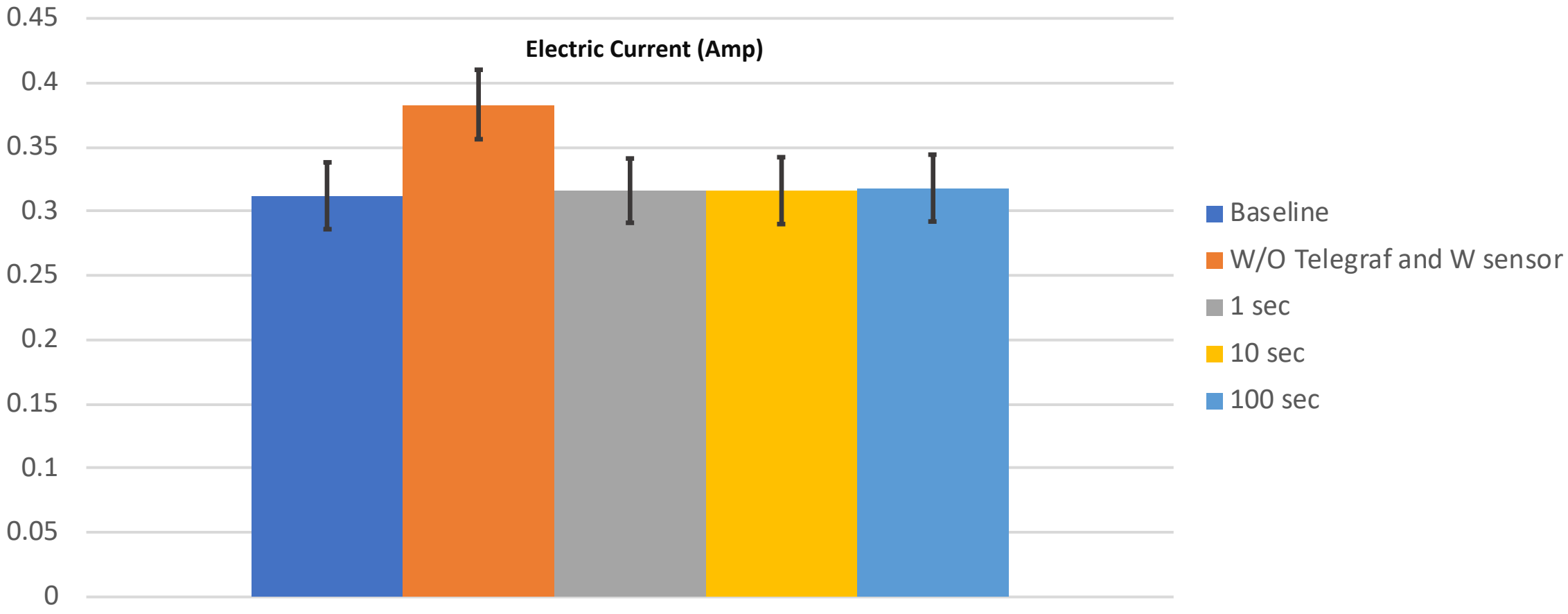
Experiment NO.	Telegraf	Sensor	Metric	Interval
3.1	-	-	-	-
3.2	-	Is Running	-	-
3.3	Is Running	Is Running	1	1 sec
3.4	Is Running	Is Running		10 sec
3.5	Is Running	Is Running		100 sec

Metric = Log parser

Sensor Overhead + Telegraf



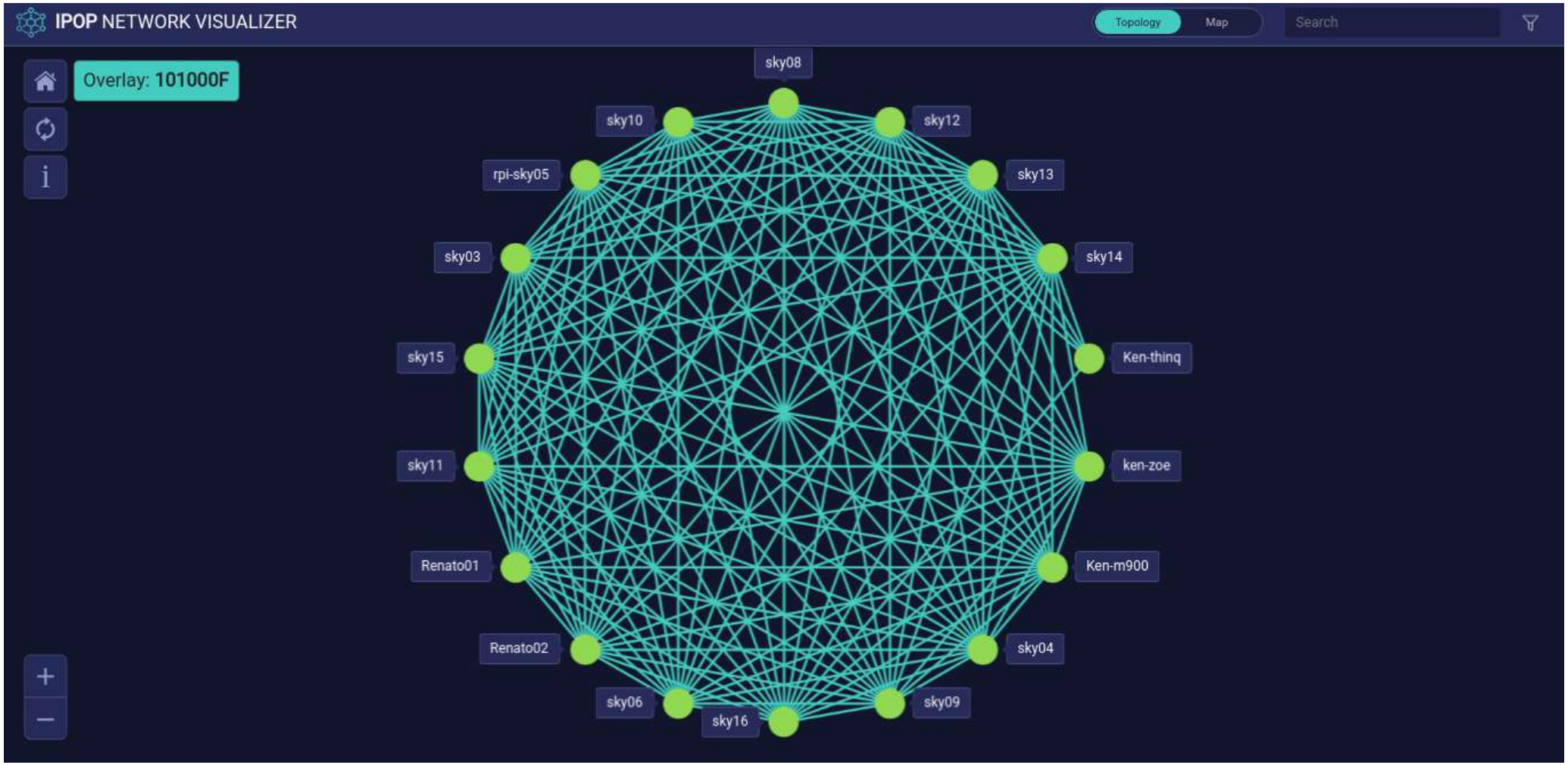
Sensor Overhead + Telegraf



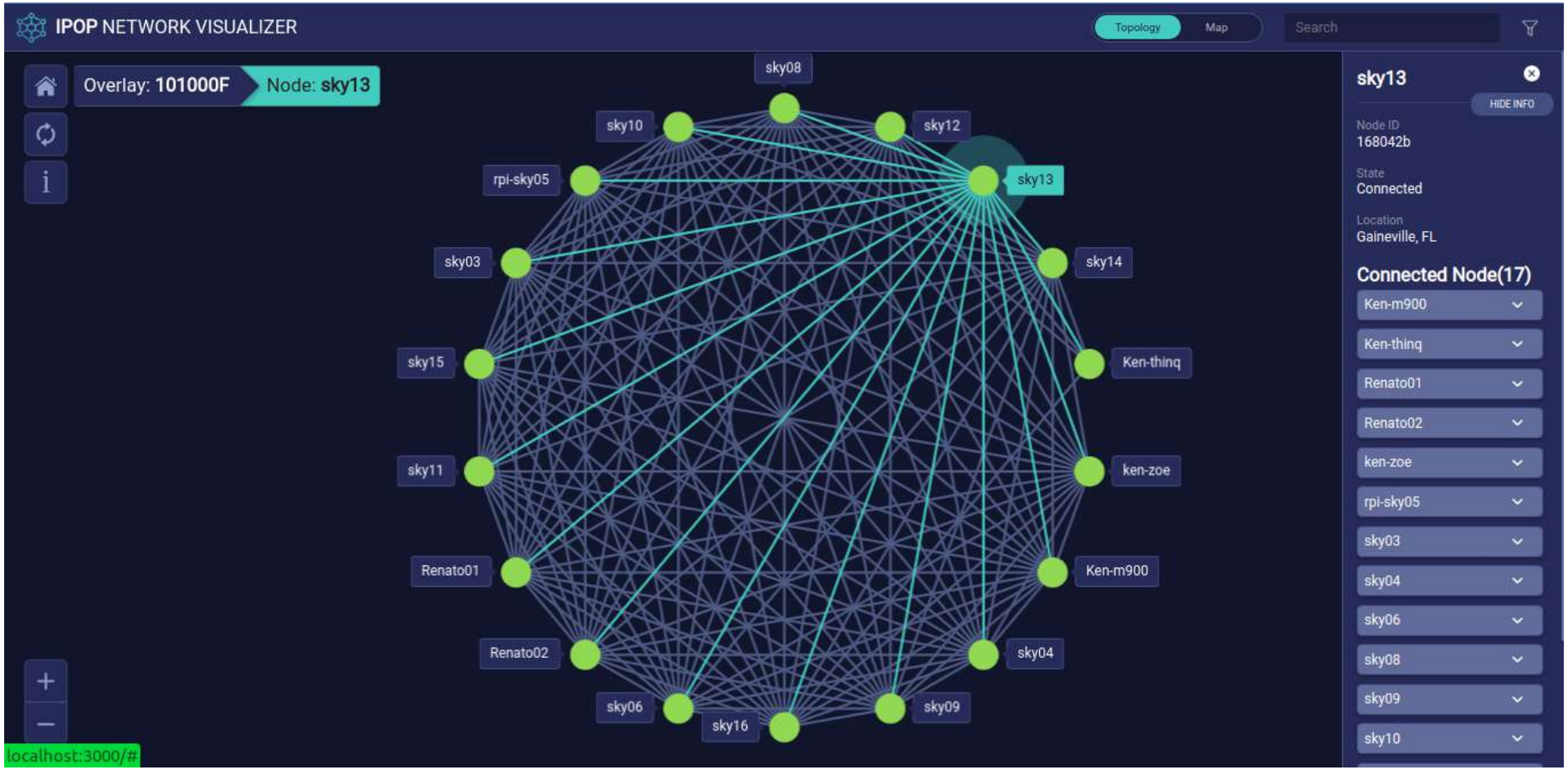
Conclusions of Experiments

- When Telegraf is running in an IoT gateway, It uses 15% more memory
- Telegraf does not use much CPU and electric current of the IoT gateway if the data polling interval is 0.1 s or above
- In general, Telegraf’s disk usage is negligible. However, sensor logging causes 4% more disk usage

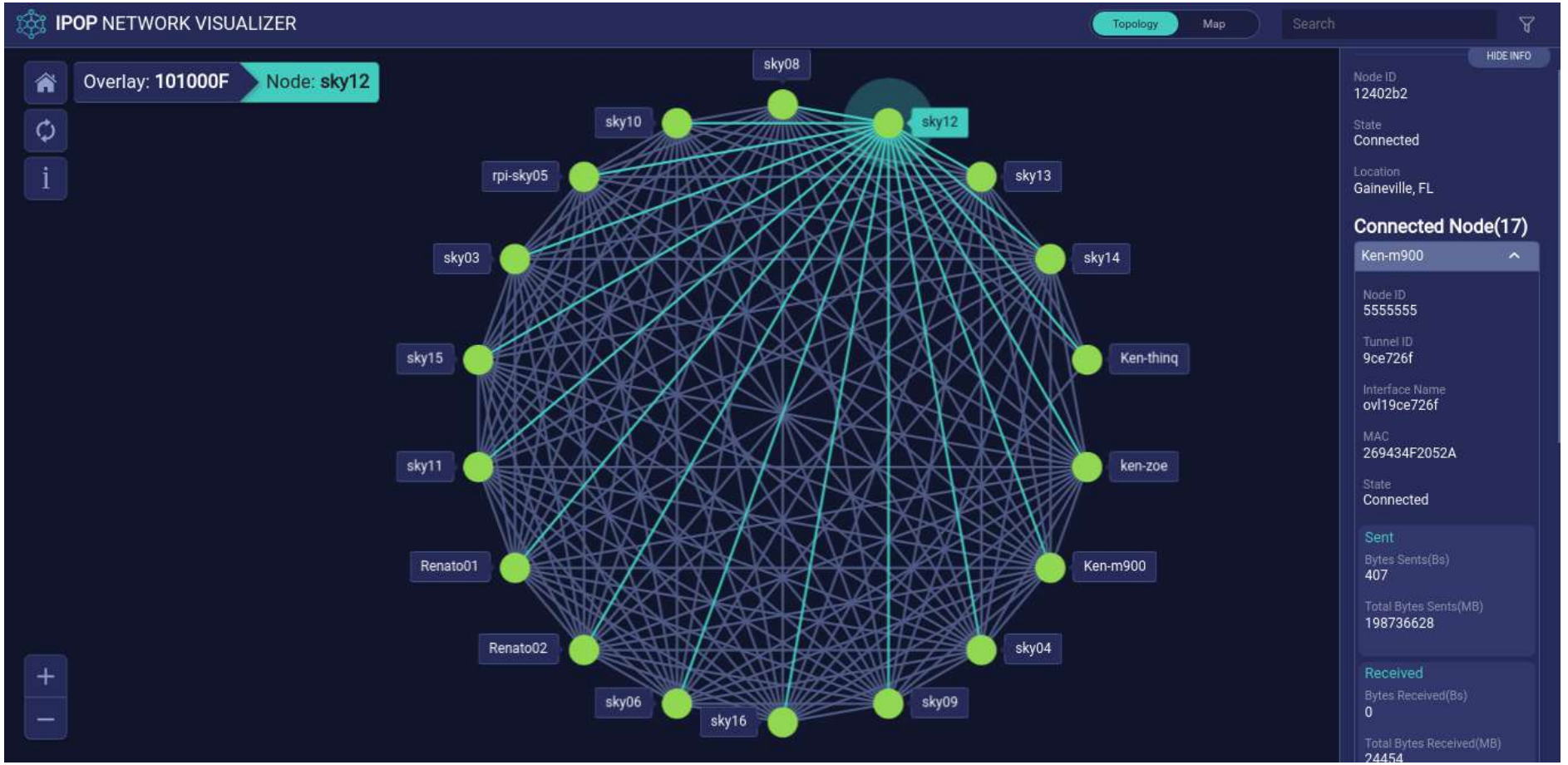
IPOP Visualizer



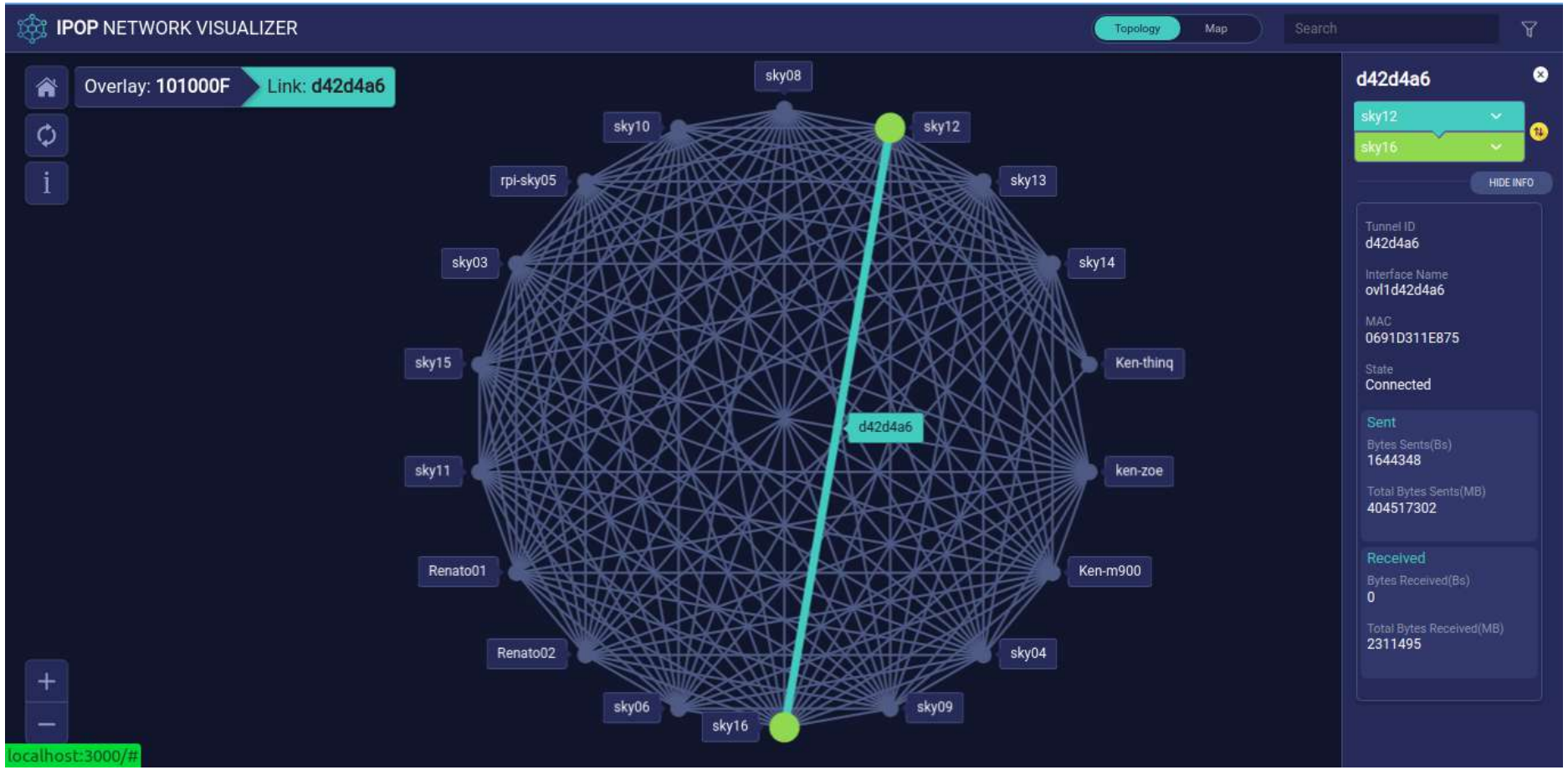
IPOP Visualizer



IPOP Visualizer



IPOP Visualizer



Conclusions

- Telegraf's scalability and low power consumption make it ideal for deployment on Raspberry Pi.
- InfluxDB is suitable for storing time-series data.

Ongoing Work

- Investigate other approaches for Telegraf to collect application data.
- Investigate the impact of other approaches on system performance.

Question ?

Monitoring Dashboard

What's in a database?

This page gives SQL users an overview of how InfluxDB is like an SQL database and how it's not. It highlights some of the major distinctions between the two and provides a loose crosswalk between the different database terminologies and query languages.

In general...

InfluxDB is designed to work with time-series data. SQL databases can handle time-series but weren't created strictly for that purpose. In short, InfluxDB is made to store a large volume of time-series data and perform real-time analysis on those data, quickly.

Timing is everything

In InfluxDB, a timestamp identifies a single point in any given data series. This is like an SQL database table where the primary key is pre-set by the system and is always time.

InfluxDB also recognizes that your schema preferences may change over time. In InfluxDB you don't have to define schemas up front. Data points can have one of the fields on a measurement, all of the fields on a measurement, or any number in-between. You can add new fields to a measurement simply by writing a point for that new field. If you need an explanation of the terms measurements, tags, and fields check out the next section for an SQL database to InfluxDB terminology crosswalk.

Source: https://docs.influxdata.com/influxdb/v1.7/concepts/crosswalk/