Introduction And Motivation
Programming IoT devices are done manually and in an ad-hoc fashion. To build custom applications, cloud services like NEST API’s are available in the market. But applications running on cloud introduces some challenges:
- Data processing latency.
- Dependency on the internet connectivity.
- Consumes a lot of bandwidth.
To overcome above challenges we need a framework that can be used to build custom applications on distributed network of IoT devices. Calvin framework can be used to build custom applications and to bring computations to the IoT devices.

IoTX Framework
IoTX framework aims at serving as an “operating system” to support the following:
- Customize user application to efficiently utilize resources from IoT devices.
- Bring some of the data analytics to the edge.
- The IoTX framework is the top most layer that provides application developers to create custom applications and to manage and deploy these applications dynamically.

Calvin Framework
- Calvin runtime handles the communication between actors and between runtimes.
- Applications are managed and run inside a runtime.
- Actor is a reusable computational unit.
- Actors communicate by passing data tokens from input/output ports.
- Multiple actors form a data flow graph which is defined as an application.

Actor Model In IoTX
- IoTX project is a layer on top of Calvin framework which will not only manage and deploy applications dynamically but also aims at serving as an efficient resource manager by using actor migration.
- Calvin exposes REST APIs.
- In this project, we have implemented an interface that communicates with Calvin running on any machine.
- IoTX uses REST APIs to control and communicate with Calvin applications.
- The framework provides a connection to the Docker containers where Calvin is installed to simulate a distributed test application environment.
- Once the user binds the actors to IoT nodes in the front-end and runs the custom distributed application, the backend collects all the data from the front-end and creates the requirements dynamically to start and run the application in Docker containers.

Data Caching
- Implemented an LRU cache in the IoTX project.
- The cache size is configurable.
- The least recently used object is the one to be removed first.

Future Work
- IoTX can be extended to implement the fault tolerance mechanism by actor migration process.
- Furthermore, IoTX can act as a scheduler to manage and assign tasks to the IoT nodes depending on the resource availability.