Introduction:

- IoT (Internet of Things) devices with limited resources are currently pushing data to cloud for data processing. We are proposing an iotX framework for reducing this dependency on cloud and utilizing resources across distributed IoT devices by carrying out lightweight processing at the Edge with dynamic task allocation.

- User is provided with a web interface for customizing IoT network topology and launching different apps to testbed to verify the desired outcome before actually implementing the setup in the real world. Thus, there is a need to store initial configuration of every entity like IoT Node, Sensor, Actuator etc. along with efficient tracking and analysis of user updates at each event.

Polyglot Persistence:

- Store basic info. (such as GUID) of IoT nodes
- Frequently used objects to reduce accessing DB.
- Store details of IoT nodes, including info. regarding attributes of nodes, links, sensors, actuators, actors and constraints.
- Provide high scalability whereas, Neo4j with its connected data or graph structure helps in tracking, analyzing network topology and Calvin apps efficiently.

Implementation:

- Calvin is a light-weight open source framework developed by ‘Ericsson Research’ team with a runtime environment for IoT devices to communicate with each other.
- Actor in Calvin framework is a set of well-defined instructions for performing actions. Calvin application is formed by set of such actors connected based on the data flow information to carry out certain function.

Calvin Application:

- Modeling IoT Network Topology: To mimic a real IoT node in our system, we have modeled each entity related to a real IoT node like Sensor, Actuator, Actor, constraints considering their fine details in MongoDB.
- When an user creates a network topology of IoT nodes, we instantiate all the resources and objects according to the user requirements. User can update the network topology which can be captured as a discrete-time event in the system using Neo4j graph database. Web interface also offers an option of accessing topology and Calvin application graph at past events to provide a flexibility of designing and utilizing the topology to the user.

Future Work:

- Using Neo4j to do pattern matching for identifying similar applications in network. Provide customized heuristics for finding optimal path between given set of nodes
- Using ‘Neo4j Doc Manager’ to fetch MongoDB documents and query them when needed to convert it into required Neo4j graph structure