### Project Goals
- To create a distributed data flow processing framework for IoT devices using Calvin framework which uses actor model and data flow programming concepts.
- Implement a lightweight controller for dynamic update of network data flow.
- Integrate the controller with the data flow processing framework for orchestrating network flow changes.

### Existing IoT Data Processing Architecture
- IoT devices offload all computational jobs to the cloud compute engine.
- This leads to increase in network latencies compared to performing computations in network.
- Real time data analytics on sensor values is carried out on cloud.
- Large number of network requests due to lack of communication and coordination among IoT devices.
- Computational power of resource constrained devices is not fully utilized.
- Most computational tasks involve low latency location awareness.

### Proposed Solution
- Reduce network overhead by making IoT devices communicate with each other through actor model of concurrency and data flow programming concepts.
- Process data as it flows through the routing nodes which enables real time data analytics.
- Actors on routing nodes perform in network computations.
- Lightweight protocol for coordination between routing nodes.
- Fault tolerance for failed routing nodes.
- Controller for orchestrating data flow among nodes.

### System Design
- Network flow is simulated using a no. actors deployed across multiple machines.
- Coordination of network routing among nodes by a data flow controller.
- Graph isomorphism check using VF2 algorithm to detect change in network flow.
- Controller schedules the new actor graph configuration and migration of the updated actor graph to assigned runtimes.
- Fault Tolerance provided by migrating actors to available runtimes.
- Kademlia DHT for maintaining peer runtimes IP addresses and ports.

### Controller Flow Update - Algorithm
1. Start controller web server in daemon mode
2. Listen for new data flow graph P via REST
3. Check for running applications
4. Check if current data flow graph G and received graph P are isomorphic
   4.1 If graphs are isomorphic, continue
   4.2 Else, if new graph vertices equal no. of active peers
   4.2.1 Parse new actor graph and create deployment script
   4.2.2 Redeploy app. with new deployment script
   4.2.3 If redeployment is successful
   4.2.3.1 Update previous actor configurations
   4.2.3.2 Kill existing application from DHT
   4.2.4 If redeployment fails, rollback

### Results
- Implemented prototype applications for triggering events on temperature changes and computing average temperature value of a sensor network.
- Implemented and tested controller for data flow updates in Odroid cluster.
- Tested application memory utilization for large number of actors while processing sensor data.

### Conclusions and Future Work
- Current approach requires redeployment of application.
- GraphQL algorithm for isomorphism check over VF2.
- Look into data flow updates through modifying the DHT.
- More fine-grained scheduling mechanism for actors on runtimes in case of failure based on network metrics.