CMPSCI 677 Operating Systems **Spring 2015**

Lab 2: April 10

Lecturer: Prashant Shenoy *Developers: Patrick Pegus II, Amee Trivedi*

System Overview

To build IoT system, supporting :

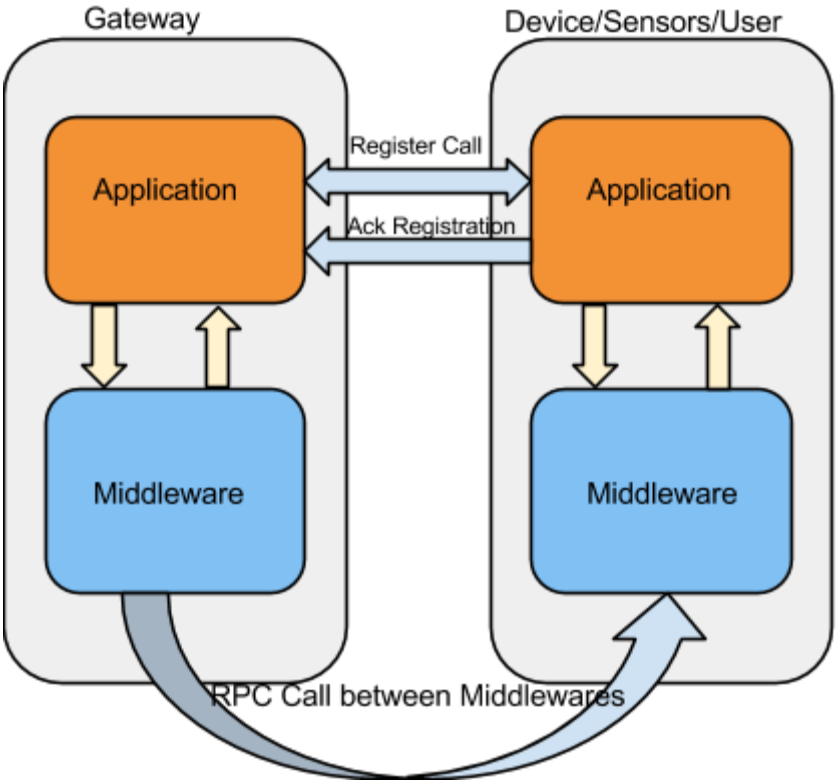
1 Gateway

3 Sensors comprising of a motion sensor, temperature sensor and door sensor

2 Devices : Smart Bulb and Smart Outlet

System Design

Programming language used is Go Lang. Go Gophers! Go!

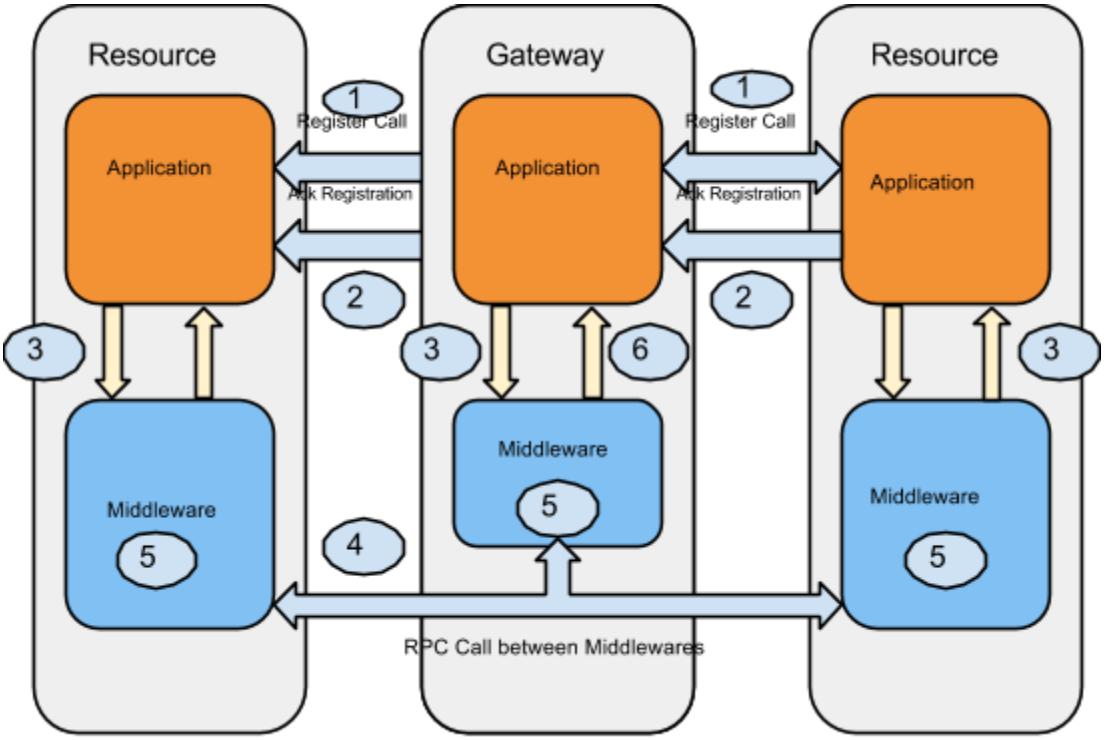


CMPSCI 677 Operating Systems **Spring 2015**

Lab 2: April 10

Lecturer: Prashant Shenoy *Developers: Patrick Pegus II, Amee Trivedi*

Interface Diagram



Description

1 -> A new device Registers, Gateway sends Device ID
2 -> Device sends Acknowledgement
3 -> Gateway sends notification to the middleware about the new device and the new device initiates its middleware
4 -> The gateway middleware sends notification to the other middlewares to update their Peertable that keeps a track of device id and Address:Port all the other middleware
5 -> Initiate leader election or berkeley time sync or clock sync algorithm based on the deviceid, timer constraints
6 -> Send notification to the Application

CMPSCI 677 Operating Systems **Spring 2015**

Lab 2: April 10

Lecturer: Prashant Shenoy *Developers: Patrick Pegus II, Amee Trivedi*

System Assumptions

1. All communication channels are up and running always. If the TCP Dial service fails, it is assumed that the device/sensor is down
2. The Gateway is the central entity of system. All intelligence is in the gateway
3. Database always comes up first, followed by Gateway and then, the rest devices.
4. The Temperature increments and decrements are always measured by unit change. i.e. Increment is +1 degree and Decrement is -1 degree
5. The Aliveness of Leader is checked every 5 seconds
6. The temperature sensor polling is done by the gateway every 60 seconds

System Design Decisions

Leader Selection Algorithm

The leader selection algorithm is "Bully Algorithm". In the given system each sensor/device is connected to the gateway. Bully Algorithm is free from starvation and deadlock this is also another reason for selection.

Database Design

Each Device/Sensor has 2 les associated with it. The event le and the state le. The primary advantage of this architecture is that querying the le for the last state of the corresponding device becomes extremely easy. The last state would be equal to the latest entry based on the timestamp.

Middleware

Each device/sensor/gateway(referred to as resource now on) has 2 layers. Application layer, which is visible to the user and the middleware layer, which keeps a track of the other middleware layers associated with peer resources. This design approach was selected to make the system more transparent and reduce code redundancy. Another, advantage is it isolates the application from middleware and so, allows middleware implementation to be easily switched

Peer Table

For the implementation of Leader Election algorithm and Berkeley Clocks it was important that each resource knows the ip address, port number of the rest peers in the system. This information

is needed by the middleware to communicate with each other, in order to facilitate notication's to their respective applications.

Berkeley Clock

Unix time was used as time-stamp in implementation of Berkeley clock.

RPC calls and non-RPC interfaces

The calls between middlewares of resources are RPC based calls. However, the calls from application to its own middleware is a non-RPC reference call.

Logical clocks over vector clocks.

Logical clocks suited out architecture more with respect to number of messages exchanged and bandwidth usage.

System Dependencies

Lane lib is used for a priority queue : github.com/oleiade/lane
UUID lib is used for logical clocks : github.com/nu7hatch/gouuid

Code Organization

All the structs, type and interface declarations are in api.go
File database.go defines the structs and interfaces needed
The middleware files are located in the subdirectory ordermw.
It mainly consists of file logical.go and BClockDummy.go. These 2 files define the structs, interfaces to gateways and RPC calls.
The rest resources are in the <resource-name> dir with main.go file, which is responsible to start the resource instance and the <resource-name>.go file that has all the interface implementations and it also starts its middleware.

Known Issue

The index out of bound error (idx) error in the home/away mode code is a known issue.