6/01/2021

IOT CIE-3

Name: Sai Sriram V
USN: 1BM18CS140
CC: 20CS5PEIOT

1) WAMP (Web Application Messaging Protocol) is a sub-protocol of Websocket which provides publish-subscribe and remote procedure call (RPC) messaging patterns. WAMP enables distributed application architectures where the application components are distributed on multiple nodes and communicate with messaging patterns provided by WAMP.

Key Features of WAMP:-

→ Transport :- For WAMP, default is Websocket
→ Session :- Session is a conversation b/w two peers.
→ Client :- Clients are peers that can have one or more roles. In publish-subscribe model client can have following roles:-
  → Publisher :- Publishes events to topics maintained by the Broker.
  → Subscriber :- Subscribes to the topics and receives the events including the payload.

In RPC model, Client has two roles:-
(i) Caller :- Issues calls to the remote procedures along with call arguments.
(ii) Callee :- Executes procedures to which the calls are routed by the caller & returns the results back to the caller.

→ Routers :- Peers that perform generic call & event routing.
In Publish subscribe model Router has the role of a Broker. and in RPC model, Router has the role of a Dealer.

→ Application Code :- It runs on the Clients (Publisher, Subscriber, Callee or Caller).

V. Sriram

Scanned with CamScanner

Name: Sai Sriram V
USN: 1BM18CS140

2b) Things Manager creates Groups, Adds appropriate member things in the network, manages member presence and makes group action easy. It benefits 3rd party application developers in three way:-

1) Application can easily collect things for a specific service by the service characteristics, not by each thing's identification.
2) Application does not require handling, tracing or monitoring many things.
3) Application does not require managing to send control messages to several things. Also, configurations and diagnostics of multiple things can be supported by this service.

2c) Amazon RDS is a web-service that allows you to create instances of MySQL, Oracle or Microsoft SQL Server in the cloud. With RDS, developers can easily set up, operate and scale a relational database in the cloud. RDS can serves as a scalable datastore for IOT systems. With RDS, IOT system developers can store any amount of data in scalable relational databases.

V. Sai Sriram

Scanned with CamScanner

Name: Sai Sriram V
USN: 1BM18CS140

4a) Python program to read from an SQS queue

```
import boto.sqs
from boto.sqs.message import Message
ACCESS_KEY = "<enter access key>"
SECRET_KEY = "<enter secret key>"
REGION = "us-east-1"
print "Connecting to SQS"
conn = boto.sqs.connect_to_region(
        REGION,
        aws_access_key_id = ACCESS_KEY,
        aws_secret_access_key = SECRET_KEY)
queue_name = 'mytestqueue'
print "Connecting to queue: " + queue_name
q = conn.get_all_queues(prefix = queue_name)
count = q[0].count()
print "Total messages in queue: " + str(count)
print "Reading message from queue"

for i in range(count):
    m = q[0].read()
    print "Message %d %s" (i+1, str(m.get_body()))
    q[0].delete_message(m)

print "Read %d messages from queue" % (count)
```

→ Python program to writing to an SQS queue:-

```
import boto.sqs
from boto.sqs.message import Message
import time
ACCES_KEY = "<enter access key>"
SECRET_KEY = "<enter secret key>"
REGION = "us-east-1"
```

V. Sai Sriram

P.T.O.

Scanned with CamScanner

Name : Sai Sriram V
USN: 1BM18CS140

```
print "Connecting to SQS"
conn = boto.sqs.connect_to_region(
    REGION,
    aws_access_key_id = ACCESS_KEY,
    aws_secret_access_key = SECRET_KEY)

queue_name = 'mytestqueue'
print "Connecting to queue : " + queue_name
q = conn.get_all_queues(prefix = queue_name)

msg_datetime = time.asctime(time.localtime(time.time()))

msg = "Test message generated on: " + msg_datetime.
print = "Writing to queue : " + msg
m = Message()
m.set_body(msg)
status = q[0].write(m)
print "Message written to queue"
count = q[0].count()
print "Total messages in queue : " + str(count)
```

V. Sai Sriram

Scanned with CamScanner

Name : Sai Sriram. V
USN : 1BM18CS140

3b) Querying a resource state using [GET]

This operation fetches the value of a simple resource. In this example, we fetch the state from the light resource.

Sequence Diagram Steps:- 1) The client application calls resource.get (..) to retrieve a representation from the resources.

2) The call is marshalled to the stack which is either running in-process or out of process (daemon).
3) The C API is called to dispatch the request
4) Where CoAP is used as a transport, the lower stack will send a GET request to the target server.
5) On the server side, the OCProcess() function (message pump) receives and parses the request from the socket, then dispatches it to the correct entity handler based on URI of the request.
6) The C++ SDK passes it up the C++ handlers associated with the OCResource.
7) The handler returns the result code and representation to the SDK.
8) The SDK marshalls the result code and representation to C++ entity handler. and from there to CoAP Protocol.
9) The CoAP protocol transports the results to the client device and the results are returned to the OCDoResource callback.
10) Results are returned to C++ client application's asyncResult callback.

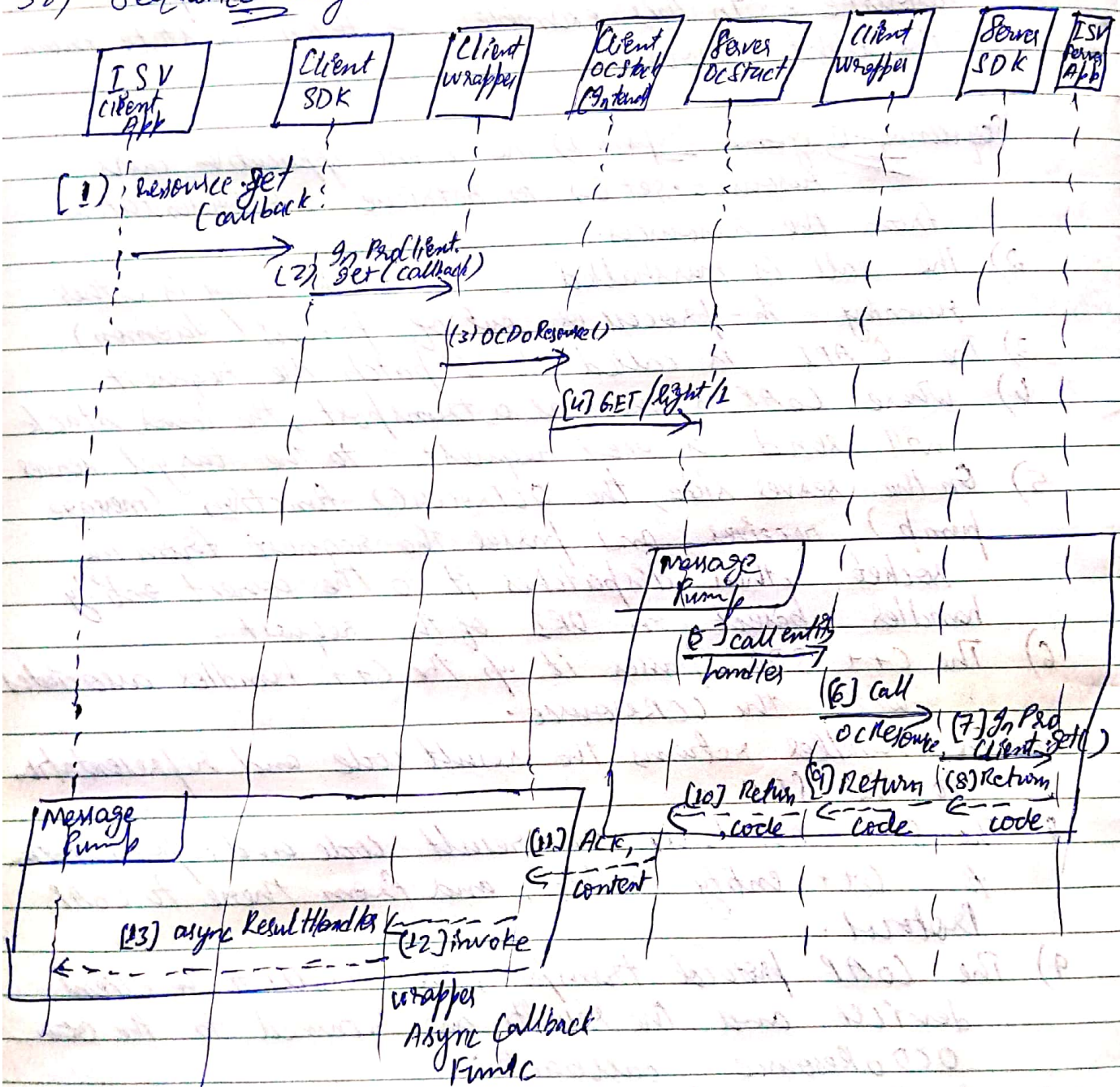
V. Sai Sriram

P.T.O

Scanned with CamScanner

Name : Sai Sriram V
VSN : 1BM18CS140

3b) Sequence Diagram :-



V. Sriram

Scanned with CamScanner

Name: Sai Sriram V
USN: 1BM18CS140

2 a) On Analyzing given diagram,
Behaviours on Resource Model:

→ Finding a resource
→ Querying, setting and observing resource state.

Here, one of the child resources on the garage door opener is the light control; it has a GET operation that allows a device to get the current light state (on/off).

V. Sai Sriram

Scanned with CamScanner