Date: 06/01/21 | IOT (CIE-3) | Name: Shreya Laddha, USN: 1BM18CS103

PART: A

1). Iotivity is an open source framework and SDK for building IOT applications.

IOTIVITY FEATURES:

* General:
  - → Core functionality is written in C for deployment.
  - → Binding available: Java, Javascript.

* Discovery & Connectivity
  - → Direct Device-to-Device
  - → Message Connectivity
  - → Supports IPV4/IPV6 on all OS
  - → Supports bluetooth serial RFCOMM
  - → XMPP hidden connectivity.
  - → Bluetooth EDR

* Resource Management:
  - → Resource Model operations: GET, PUT, cancel, notification etc.
  - → CBOR encoding & Decoding.

* Services:
  - → Soft Virtual Sensor Manager, Protocol Plugins, Thing, Notification, Control.

* Target: Generic Linux, Tizen, Yocto, Android

* Source code is managed in Yocto review Server.

Name: Shreya Laddha
USN: 1BM18CS103

PART : B

2a). This operation sets the value of a simple resource. In the ~~dt~~ sequence diagram given, ~~@~~ the service provided is ~~on~~ to turn on a light resource and sets its brightness ~~to~~ 30%.

Steps:

1. The client application calls resource.put(attributeMap, callback) to set representation of resource.
2. client sdk internally calls the setResourceAttributes function to the client wrapper using inProcClient.setResourceAttributes.
3. Sent PUT request to remote device.
4. The OCProcess() service reads the packet from the server and dispatches the request for the URI given.
5. The entity handler, in this case C++ API passes the results to the upper layer handler. The results are processed in the entity handler.
6. The entity in the upper layer by the app developer is invoked.
7. The entity returns success or failure as a response.
8. Refer Steps 8, 9, 10 returns ~~suc~~ response to lower layer from client.

Shreya

Name: Shreya Laddha
USN: 1BM18CS103

11. Result is formatted and sent over network to client.
12. The OCProcess() service reads results via the callback passed to OCDoResource.

2b) Amazon EC2:

→ EC2 : Elastic Compute Cloud.
→ It is an infrastructure-as-a-Service.
→ A Web service that provides computing capacity in the form of VM's.

Amazon's Autoscaling:

→ Autoscaling allows automatically scaling Amazon EC2 capacity up or down according to user conditions.

→ Autoscaling along with Amazon EC2 allows automatically scaling the capacity up or down and increase the number of EC2 instances in the application during spikes in workload and scales down the capacity when workload is low to save cost.

→ The settings for Autoscaling group include maximum and minimum number of instances.

Name: Shreya Laddha
USN: 1BM18CS103

2e) WAMP:

→ It is ideal for distributed, ~~multicast~~ multi-client and server applications.

→ Such as multi-user driven business applications, sensor networks (IOT), MMOG's

→ CONCEPTS OF WAMP

1] Transport: Channel that connects two peers and is bidirectional. Default transport is WebSocket

2] Session: Conversation between 2 peers over a transport

3] ~~Transport~~. Roles of Client: [clients are peers].

> Publisher: publishes event to the topic by Broker

> Subscriber: Subscribes to these topics.

4] RPC Model client roles:

> Caller: issues calls to remote procedures with call arguments

> Callee: executes the procedures to which the Caller issues the calls and returns the result.

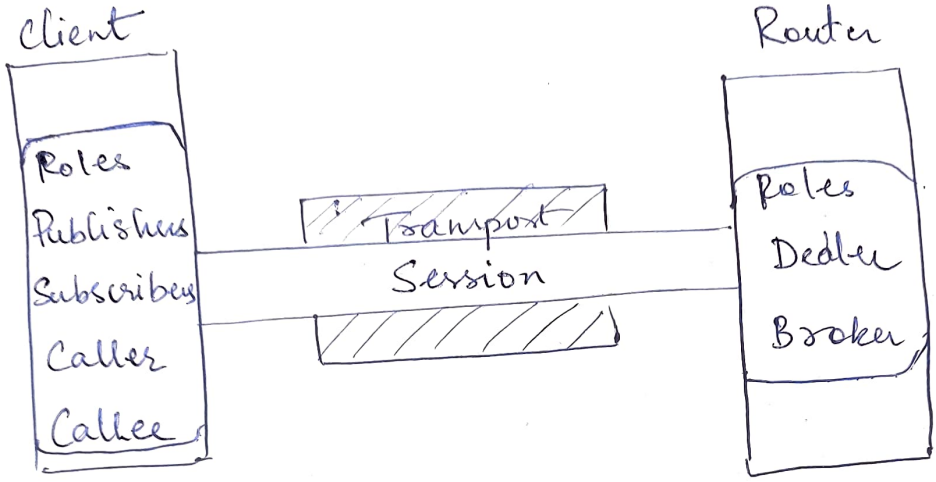
5] Router: Peers that perform generic calls

→ Broker: Acts as a router and routes messages

→ Dealer: Acts as a router and routes RPC calls

Name: Shreya Laddha
USN: 1BM18CS103

6] Application Code: Runs on the Client.

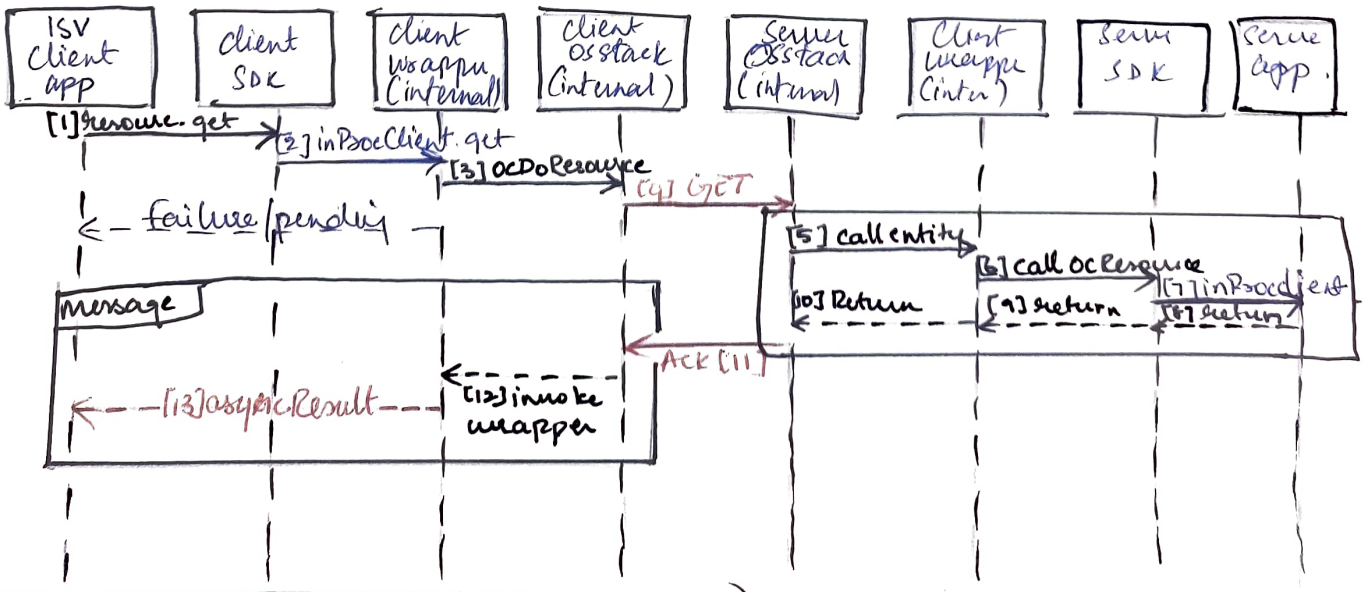


PART : C

3a) Querying Resource State: [GET]

> fetches values of simple resource.

SEQUENCE DIAGRAM:



Name: Shreya Laddha
USN: 1BM18CS103

Steps:

1. The client application calls resource.get to retrieve a representation from the resource.
2. Call is marshalled in the stack (daemon).
3. The C API is called to dispatched the request
4. CoAP is used as the transport, lower stack sends a GET request.
5. OCProcess() function recieves request from socket.
6. C++ entity handler parses the payload on the server stack.
7. C++ SDK passes it to C++ handler with OCResource.
8. The handler returns the resource code to the SDK
9. SDK marshals the result code to the ~~CoAP protocol~~ C++ handler
10. Entity handler returns result code to the CoAP.
11. CoAP protocol transports the result to client device
12. Results are returned from OCDoResource callback.
13. Results are returned to the C++ client application asyncResultCallback.

Shreya

Name: Shreya Laddha
USN: 1BM18CS103

4a]. Program to launch an EC2 instance

```
import boto.ec2
from time import sleep

ACCESS_KEY = " enter access key"
SECRET_KEY = " enter secret key"

REGION = " us-east-1"
AMI_ID = " ami-00f89009"
EC2_KEY_HANDLE = "enter key handle "
INSTANCE_TYPE = "t1.micro"
SECGROUP_HANDLE = " default "
print "Connecting to EC2"
conn = boto.ec2.connect_to_region(REGION, aws_access_key_id
                                  = ACCESS_KEY,
                                  aws_secret_access_key=
                                  SECRET_KEY)

print "Launching instance with AMI-ID %s, with keypair %s
instance type %s, security group %s" %(AMI_ID, EC2_KEY_HANDLE,
                                       INSTANCE_TYPE, SECGROUP_HAND

//LAUNCH
reservation = conn.run_instances(image_id=AMI_ID,
                                 keyname = EC2_KEY_HANDLE,
                                 instance_type = INSTANCE_TYPE,
                                 security_group=[SECGROUP_HANDLE,])

instance = reservation.instances[0]
print "Waiting for instances to be up & Running "
status = instance.update()
while status == 'pending':
    sleep(10)
    status = instance.update()
```


Shreya

Name: Shreya Laddha
USN: 1BM18CS103

```
if status=='running':
    print("\n Instance is now running. Instance Details are: ")
    print "Instance size: "+str(instance.instance_type)
    print("Instance state: "+str(instance.state)
    print "Instance launch time: "+str(instance.launch_time)
    print "Instance public DNS: "+str(instance.public_dns_name)
    print "Instance private IP: "+str(instance.private_ip_address)
```

Program to stop EC2 instance.

```
import boto.ec2
from time import sleep
ACCESS_KEY = "enter access key"
SECRET_KEY = "enter secret key"
REGION = "us-east-1"
print "Connecting..."
conn = boto.ec2.connect_to_region(REGION, aws_access_key_id
                                  = ACCESS_KEY,
                                  aws_secret_key = SECRET_KEY)

print("Getting all running instances"
reservation = conn.get_all_instances()
instanceid = instance_rs[0].id
print "Stopping instance with ID: "+str(instanceid)
// STOP
conn.stop_instances(instance_ids=[instanceid])
status = instance.update()
while not status=='stopped':
    sleep(10)
    status = instance.update()
print("Stopped Instance")
```