Lab report

Service-Oriented Approaches for the IoT

Realized by

DRAOUI Bilal

Supervised by

Dr.Nicolas Ferry

We will first start by introducing the technologies that we will be working with in this:

Node.js: An open-source, cross-platform, asynchronous event-driven JavaScript runtime environment. It is designed to build scalable network applications.

TypeScript: A strongly typed programming language that builds on JavaScript, giving you better tooling at any scale.

Express.js: Or simply Express, is a back end web application framework for building RESTful APIs with Node.js

Node-coap: An OPEN Open Source Project, that facilitates working with the COAP protocol

1- Simple client-server

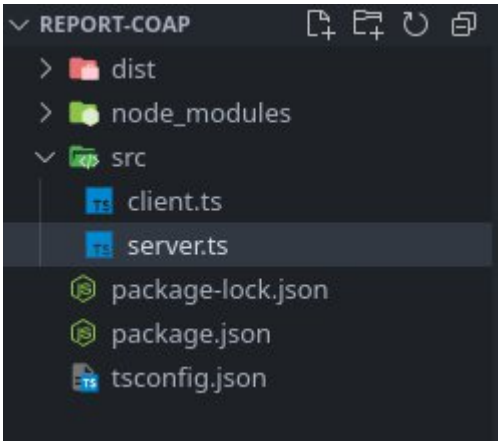
The first thing we do is initialize the node project:

```
~/Downloads/projects/web/report-coap  npm init -y
Wrote to /home/drale/Downloads/projects/web/report-coap/package.json:

{
  "name": "report-coap",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Then we install the necessary dependencies:

```
~/Downloads/projects/web/report-coap  npm install coap

added 21 packages, and audited 22 packages in 677ms

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
~/Downloads/projects/web/report-coap  npm install -D typescript nodemon

added 32 packages, and audited 54 packages in 1s

7 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
~/Downloads/projects/web/report-coap
```

after this, we create 2 files that will represt our client and our server :



In our server file:

```
import * as coap from 'coap'

const server = coap.createServer()

server.on('request', (req: coap.IncomingMessage, res: coap.OutgoingMessage) => {

    const url = req.url.split('/')[1]
    if (url === 'time') {
        res.end(`{"time":"${Date.now()}"}`)
    }
    else {
        res.end('404')
    }

})

// the default CoAP port is 5683
server.listen(() => {
    console.log('server started on port 5683')
})
```

In our client file:

```nodejs
import * as coap from 'coap'

const req = coap.request({
    pathname: 'time',
})

req.on('response', (res: coap.IncomingMessage) => {
    const json = JSON.parse(res.payload.toString())
    console.log(json)
    res.on('end', () => {
        process.exit(0)
    })
})

req.end()
```

So when we start the server and then execute the client we get the current time back in epochs:

```
~/Downloads/projects/web/coap  git main  cd ../report-coap
~/Downloads/projects/web/report-coap  npm run watch
[4:10:22 AM] File change detected. Starting incremental compilation...

[4:10:22 AM] Found 0 errors. Watching for file changes.
```

report-coap : npm run server

```
~/Dow/p/web/report-coap  npm run server

> report-coap@1.0.0 server
> nodemon ./dist/server.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./dist/server.js`
server started on port 5683
```

report-coap : npm run client

```
~/Dow/p/web/report-coap  npm run client

> report-coap@1.0.0 client
> nodemon ./dist/client.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./dist/client.js`
{ time: '1665886228282' }
[nodemon] clean exit - waiting for changes before restart
```

2- Simple client-server

Sometimes we need to send a payload along with our request to coap server, so we will simulate that by sending the type of time that we want (epoch or ISO)

We add the following lines to our server code:

```
server.on('request', (req: coap.IncomingMessage, res: coap.OutgoingMessage) => {

    const url = req.url.split('/')[1]
    const payload: Payload = JSON.parse(req.payload.toString())

    if (url === 'time') {
        switch (payload.type) {
            case PayloadType.EPOCH:
                res.end(`{"time":"${Date.now()}"}`)
                break;

            case PayloadType.ISO:
                res.end(`{"time":"${new Date().toISOString()}"}`)
                break;

            default:
                break;
        }
    }
    else {
        res.end('404')
    }

})
```

And we also add this to our client code:

```
req.write(JSON.stringify({"type": PayloadType.ISO}))
```

So now when we run the code we get the following result:

```
[4:29:07 AM] File change detected. Starting incremental compilation...

[4:29:07 AM] Found 0 errors. Watching for file changes.
```

```
report-coap : npm run server
~/Dow/p/web/report-coap  npm run server

> report-coap@1.0.0 server
> nodemon ./dist/server.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./dist/server.js`
server started on port 5683
```

```
report-coap : npm
~/Dow/p/web/report-coap  npm run client

> report-coap@1.0.0 client
> nodemon ./dist/client.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./dist/client.js`
{ time: '2022-10-16T02:29:31.806Z' }
[nodemon] clean exit - waiting for changes before restart
```

3- Timeout

Sometimes a server might take too long to respond to a client, so the latter must have a timeout mechanism so that it will not stay stuck waiting for the server.

In order to simulate this, we add a timeout to the server of 250 seconds:

```
server.on('request', (req: coap.IncomingMessage, res: coap.OutgoingMessage) => {
    setTimeout(() => {

        const url = req.url.split('/')[1]
        const payload: Payload = JSON.parse(req.payload.toString())

        if (url === 'time') {
            switch (payload.type) {
                case PayloadType.EPOCH:
                    res.end(`{"time":"${Date.now()}"}`)
                    break;

                case PayloadType.ISO:
                    res.end(`{"time":"${new Date().toISOString()}"}`)
                    break;

                default:
                    break;
            }
        }
        else {
            res.end('404')
        }
    }, 250 * 1000)
})
```

And on the client, we add the listener on the timeout event. And another thing that we need to do is to not make the request confirmable, so that the client will not wait for the server:

```
const req = coap.request({
    pathname: 'time',
    confirmable: false
})

req.write(JSON.stringify({"type": PayloadType.ISO}))

req.on('timeout', (err) => {
    console.log('timeout')
    process.exit(1)
})
```

And after we run the program we see the following result:

```
[4:53:34 AM] File change detected. Starting incremental compilation...

[4:53:34 AM] Found 0 errors. Watching for file changes.
```

report-coap : npm run server

```
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./dist/server.js`
server started on port 5683
[nodemon] restarting due to changes...
[nodemon] starting `node ./dist/server.js`
server started on port 5683
[nodemon] restarting due to changes...
[nodemon] starting `node ./dist/server.js`
server started on port 5683
[nodemon] restarting due to changes...
[nodemon] starting `node ./dist/server.js`
server started on port 5683
[nodemon] restarting due to changes...
[nodemon] starting `node ./dist/server.js`
server started on port 5683
[nodemon] restarting due to changes...
[nodemon] starting `node ./dist/server.js`
server started on port 5683
```

report-coap : npm

```
> nodemon ./dist/client.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./dist/client.js`
{ time: '2022-10-16T02:29:31.806Z' }
[nodemon] clean exit - waiting for changes before restart
[nodemon] restarting due to changes...
[nodemon] starting `node ./dist/client.js`
[nodemon] restarting due to changes...
[nodemon] starting `node ./dist/client.js`
[nodemon] restarting due to changes...
[nodemon] starting `node ./dist/client.js`
[nodemon] restarting due to changes...
[nodemon] starting `node ./dist/client.js`
[nodemon] restarting due to changes...
[nodemon] starting `node ./dist/client.js`
timeout
```

4- Observable

With observable, a client can keep receiving data from the server, to enable it we modify our server code:

```
server.on('request', (req: coap.IncomingMessage & { options: { name: string, value: any }[] }, res: coap
    const isObservable = req.options.find((option) => option.name === 'Observe')

    const url = req.url.split('/')[1]
    const payload: Payload = JSON.parse(req.payload.toString())

    if (url === 'time') {

        switch (payload.type) {
            case PayloadType.EPOCH:
                if (isObservable) {
                    setInterval(() => {
                        res.write(`{"time": "${Date.now()}"}`)
                    }, 100)
                }
                else {
                    res.end(`{"time": "${Date.now()}"}`)
                }
                break;
```

And we modify our client as follows:

```
const req = coap.request({
    pathname: 'time',
    observe: true,
})
```

(alias) enum

And we get the following result once we execute the code:

```
[5:16:04 AM] File change detected. Starting incremental compilation...

[5:16:04 AM] Found 0 errors. Watching for file changes.
```

report-coap : npm run server

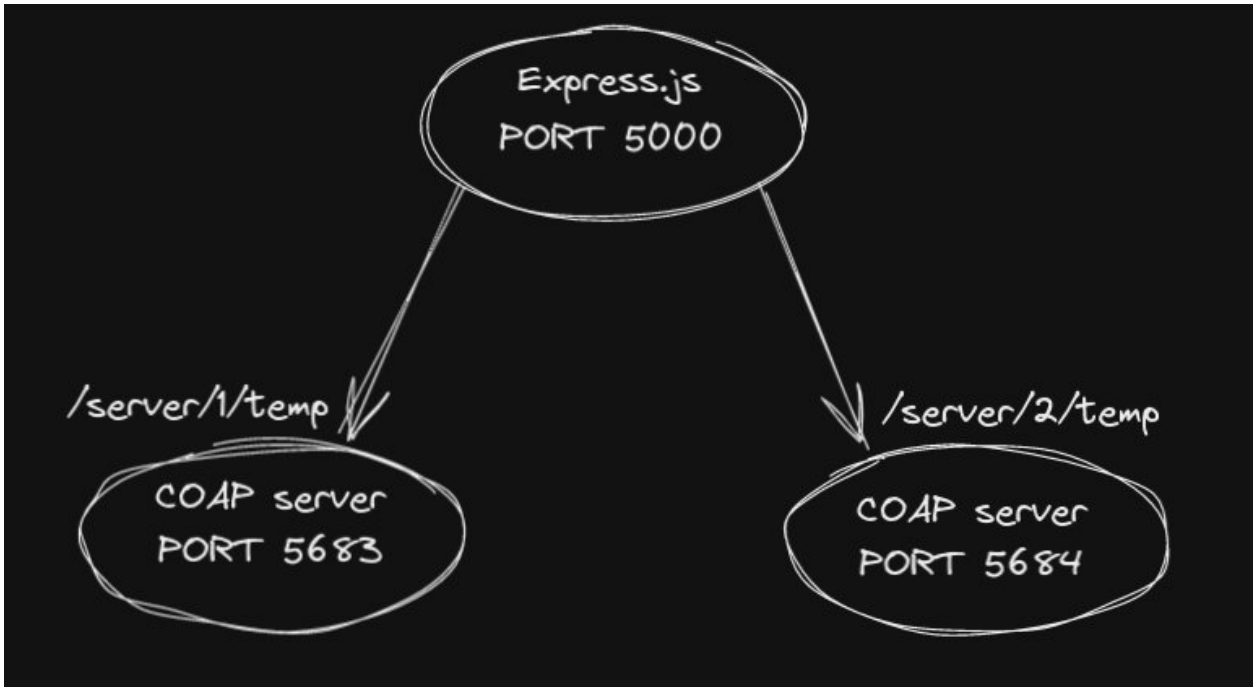
```
~/Dow/p/web/report-coap  npm run server

> report-coap@1.0.0 server
> nodemon ./dist/server.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./dist/server.js`
server started on port 5683
```

report-coap : npm run client

```
~/Dow/p/web/report-coap  npm run client

> report-coap@1.0.0 client
> nodemon ./dist/client.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./dist/client.js`
{ time: '2022-10-16T03:14:00.533Z' }
{ time: '2022-10-16T03:14:00.533Z' }
{ time: '2022-10-16T03:14:00.533Z' }
{ time: '2022-10-16T03:14:00.533Z' }
{ time: '2022-10-16T03:14:00.533Z' }
{ time: '2022-10-16T03:14:00.533Z' }
{ time: '2022-10-16T03:14:00.533Z' }
{ time: '2022-10-16T03:14:00.533Z' }
{ time: '2022-10-16T03:14:00.533Z' }
{ time: '2022-10-16T03:14:00.533Z' }
```

5- Bridge

In this last section, we will make an interface to interact with our coap servers using the famous RESTAPI library **express**. We will have the following architecture:



You can see the source code for this architecture in the following git repository:

https://github.com/DraouiBilal/Coap-Express

Or you can try it in the following website:

https://coap-express.vercel.app/