How to transfer data from an embedded node server to a remote node server with socket.io

In class we learned how to create dynamic web pages with socket.io by creating web-socket connection between the embedded node server and the web client (front-end Javascript). Now what if you want a node server running in the beaglebone to communicate with a node server hosted in a remote server (such as the SFU VM)? One easy way to accomplish this is by RESTful protocol but this is quite slow [1] and thus is not suitable for real time applications. A better solution would be using websockets and this guide will show you how to do that.

Dependencies

You need internet connection on the beaglebone and the following npm packages:

- socket.io
- socket.io-client

The embedded server

The embedded server acts as a socket client that connects to the socket server. It triggers the server event along with the data it wants to send and may have an event of it's own in case of any reply from the server. In this case 'compute' is the server event and 'result' is the client event.

```
//some server setup code

const REMOTE_SERVER_URL = 'http://192.168.7.1:3000/';
//the 'http://' needs to be included otherwise the server event won't be triggered!
//PS: URL will be different if not connecting through ethernet over usb

var socketClient = require('socket.io-client').connect(REMOTE_SERVER_URL, {
       reconnection: true
});

socketClient.on('result', function (data) {
       console.log('result = ' + data.result);
       io.sockets.emit('displayResult', data);
});

var num = 0;
setInterval(function() {
       socketClient.emit('compute', {number: num});
       num = num + 1 % 50000; //prevent overflow
}, 1000);
```

The remote server

The remote server acts as a socket server, listening (and perhaps responding with data) events that are triggered by the socket client.

```
//some server setup code

server.listen(3000);

var io = require('socket.io').listen(server);
io.on('connection', function (socket) {
	socket.on('compute', function (data) {
		console.log('number = ' + data.number);
		socket.emit('result', {
			result: (data.number+3)
		});
	});
});
```

Sample Code Explanation

The architecture above simply adds a number by 3 on the server and return the result to the client. The embedded server emits the remote server event `compute` every 1 second with an increasing number (capped at 50k) and has it's own `result` event that simply displays the result on the screen ('displayResult' event on client side javascript). The remote server simply waits for its compute event to be triggered and trigger the result event on the embedded server.

Full Code

The full code is in 'test-socket' folder in the zip package.

References

[1] http://blog.arungupta.me/rest-vs-websocket-comparison-benchmarks/
[2]
https://stackoverflow.com/questions/8837236/how-to-connect-two-node-js-servers-with-websockets