## Intro

As part of our simulation framework we will allow an application developer to upload an application bundle as well as RDTS which these apps make use of. The general requirement is to have the application deployed to all the devices within our application. The RDT library that the application makes use of should be deployed to all devices in our simulation.

## Plan

We plan to allow an application developer who is registered in a simulation to be able to view available applications already deployed in the simulation and the option to upload a new one. The application uploaded must be packaged within a folder and can contain things like css, html and  javascript, as these will be HTML5 applications. Our application framework will specify that an app must contain a javascript file at minimum.  At this stage the developer or user of the simulation is invited to chose from a list of RDT's to attach to the application. This is simply a reference to the RDT that exists on the server. The user should also be able to upload a new RDT if necessary.

At the application registration phase, the developer will be given the option to deploy the application to all the devices in the simulation. This will replicate the RDT across all the devices in the simulation.

After application deployment and registration the application should be viewable on all devices in the simulation. An application is viewable via a link or an iframe displaying the HTML5 page.

Within the device view we believe that the user should be able to alter the value of the RDT by manipulating the element the RDT is listening on the application.

## Remarks and Justification

We believe that attaching an RDT to an application should include referencing the RDT on the device. Since the application has been deployed on the server, it would be best to have the application simply require the RDT. In order to attach an RDT to an application,  the developer should ensure that the application is fully compatible with the RDT which they aim to have it use, that is, that calls within the application are being sent to a specific RDT.

Question:
How do you ensure that, when a device is looking at an application, they are able to affect the RDT. That is when the user is looking in an application and clicks the increment button, how do we ensure that the event of clicking on that button will actually affect the RDT on the server. Because RDT's and Apps are loosely decoupled there must be a way to bind an event within the application to manipulating the RDT?

## Suggestions

One way - and easily accomplished - is to have the App already contain not only HTML5 but also the  RDT that it wishes to use. The application developer is in charge of making sure the correct elements reference the RDT(s) they wish to use. Our Environment simply takes the RDT in the app wants to use and replicates across the devices. When a device looks into an app, they value it sees is the value from the RDT it holds. The application developer is in charge of making sure the app is correctly bundled with calls the correct RDTs. The simulation environment takes a specified RDT, puts it on the server, replicates it and ensure that when the application tries to manipulate the RDT, the RDT on the server is updated. The HTML5 page is sent a new copy of the RDT whenever an event on the RDT occurs in one of the devices using the simulation.

## Explanation

Much like how our simulation is set up, the actual rdt and application are stored on the server. What the device sees is a view into the simulation running on the server.

The application uploaded should contain the following things
An index.html for rendering the html view
A main.js file for any working javascript
An app.json file which will contain the following information:

This JSON file is used to identify the requirements of the application. The file will be of the

following format:
app-name: {the name of this application}
default-page : {what is the default page for this application. Usually index.html}
rdts_used : { a list of names of the rdts the application uses}

This will be used to identify the unique requirements of each app. It also allows us to support multiple RDTs per device. When the HTML5 application is deployed to all the devices, a copy if each RDT this application specified in this app.json is given to each software device on our server using the attachRDT function. We will check the names of the rdt in the list to the ones we have stored on our server and make sure that they match. This also means that the rdts must have unique names. As long as we check against clashed when the simulation manager uploads a server side RDT this should be fine.

There is only one other thing we would require this application developer to have include in their HTML5 app.
If they wish to use our simulation to test replication of this data type, any event binding to an RDT must call a function called : manipulateRDT({rdt_name}, {rdt_function}).  It takes in the name of the rdt to be manipulated and the type of manipulation to be done on the RDT.

We want this because applications could have multiple RDTs. Our simulation framework should be able to handle any RDT manipulation within the local device's HTML5 application correctly and uniformly. This manipulateRDT() function is implemented in the simulation framework. The developer doesn't need to know how it will manipulate the RDT. Once the function is called we would retrieve the local device id calling the function, the simulation_id that device is in and ask for our server side simulation manager to manipulate said RDT for the device (using socket.io ). Once this has been executed on our server side simulation the RDT is passed down to the client (or local device) and this manipulateRDT would return this new RDT.

so any event manipulating a designated on the app should include a snippet that would look like:
counterRDT = manipulateRDT('counterRDT', 'incr');
Here counterRDT is the RDT the application developer is using.  The app.json file would include this RDT name in the list of RDTs used.

Our function would retrieve the device_id calling the function and simulation_id and send the request to the simulation on the server. The return value is the new RDT that is attached to the software device on the server after it has been manipulated.

So theoretically any application can use any number of RDTs as long as they:
1.) Tell us which RDTs their application uses and identify their application through the app.json file. The RDT name(s) must match the name of the RDT in our framework.
2.) Bind events performed on the application with the manipulateRDT function. All they need to do is pass in the RDT name to be manipulated and the name of the function to be called, our simulation manager on the server deals with everything else and returns to this local device their new rdt. This way we can know WHICH RDT the application is trying to manipulate and HOW they want to manipulate it.