**IoT Resource Tree Representation(RTR) for Massively Large Data Sets Project Proposal**

**1. Vision and Goals of the project:**

IoT(Internet of things) is a promising new technology. It may link all the embedded computing devices using traditional internet infrastructure. The IoT RTR UI is a user-friendly platform to collect, manage and visualize mass quantities of data. Goals of IoT RTR:

* A webapp to view 1 million+ data points collected from the IoT. Visual aids must give the user a sense for extracting meaning from these mass quantities of data in an intuitive way.
* Optimize the client-side response time in the web browser to provide a fluid user experience.
* Implement management functions for adding or deleting nodes from the tree.

User/Personas Of the Project

The IoT is still in its infancy, and the industry is still trying to figure out how to best package and present this mass quantity of data. Our project, while utilizing industry standards for IoT systems, seeks to become a prototype that will be used as a model for further integration as the IoT evolves.

This tool is designed to be a webapp hosted on a website, or can be executed locally as local server. The users will be companies and individuals with large sets of data who would like to extract meaning through visual aids as well as manage their data.

**2. Scope and Feature of the Project:**

RTR-UI

* Present a simple and compelling front-end interface for end-users
* Client-Side Render - Use the standard D3(JQuery) library to render the data clientside in the web browser
  + Multiple Views - Different visualizations will be used depending on the amount of devices and data points in the database. As the user narrows down the area of focus on the data, new visualizations will be used to better inform the user of the data’s significance. (described more later)
    - Grouped Bar Chart
    - Zoomable Sunburst Model
* Scalability – Must scale to potentially 1 million+ devices/datapoints
  + Must optimize to keep client-side response time low
* Data Management – User should be able to modify the database
  + Add or Delete Devices
  + Add or Delete Data points
* Server-Side Processing – Processing of data must be done in the cloud rather than in the client in order to keep the response time low.
  + Data will need to be cached to not waste resources downloading the same data.
  + Sort the data and make it qualified for the D3(JQuery) API.

§ Extract only relevant information from the database to build a second more optimized database.

§ Use the ontologyRef to extract data type, so that we can categorize and block data together in the UI by the type of data they represent.

§ If we can identify ip of devices we can filter/sort data by geographical region. Continent->Country->State->City.

§ Perhaps user only wants data from a certain year->month->day->hour->minute or date\_range which will lead to a smaller data set and potential for different visualization aid to be used to better interpret the data.

**3. Solution Concept**

Global Architectural Structure Of the Project:

This project has been previously implemented with a single tree, the treewalker, displaying 1M+ nodes which shows great difficulty in retrieving the data. Additionally, the data is rather meaningless as one might have to scroll through 1M leaves in a web browser to find a specific point or device.

Our Models for Visualizing Big Data

Our concept is to implement several data models in different layers. The first layer can be implemented with a grouped bar chart (<http://bl.ocks.org/mbostock/3887051>) **.** In this way, the problem in the old model which the data cannot be quantified in a meaningful way is solved.

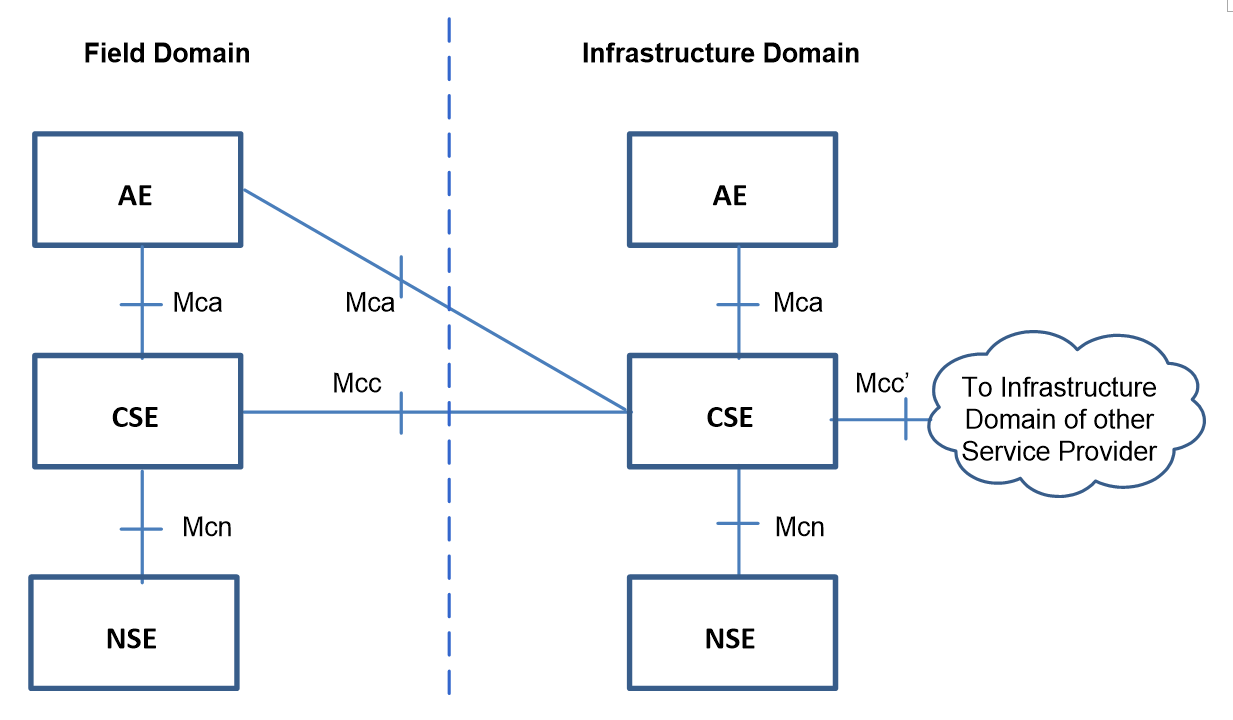
The 1 M+ nodes thus can be scaled down to a certain degree depending on the container size of a certain range of data. For example, temperature can be used in one “group block” with each bar representing the quantity of nodes within a certain range of temperature. This gives a good view of the distribution of values for a certain date type, when dealing with mass amounts of data that are otherwise difficult to quantify as a whole.

However this model does make some assumptions on the distribution of data that we need to talk to the customer about. If for example, we have 25 or less types of data then each data type can be allocated to one group. Then it doesn’t matter how many data points we have in those groups as they are covered by the bar graph for the corresponding data types group. If however there are 1,000+ types of data then there would need to be 1000 groups which would make this visualization very impractical as you would need to sidescroll to find the particular data type.

In addition (or as a replacement) for the above we can utialize a zoomable sunburst model (Coffee Flavor Wheel, <http://www.jasondavies.com/coffee-wheel/> ) or a sequence sunburst model (<http://bl.ocks.org/kerryrodden/raw/7090426/>) can be used. This kind of structure can reduce the data size quickly and provide a clear view of the nodes. Several partition methods through meta data (ie. location, date/time, or data\_type) are needed for this model.

When the nodes are reduced to a quantity of several hundred, a normal expandable tree structure can then be implemented for checking/modifying individual node information on the third layer of our model.

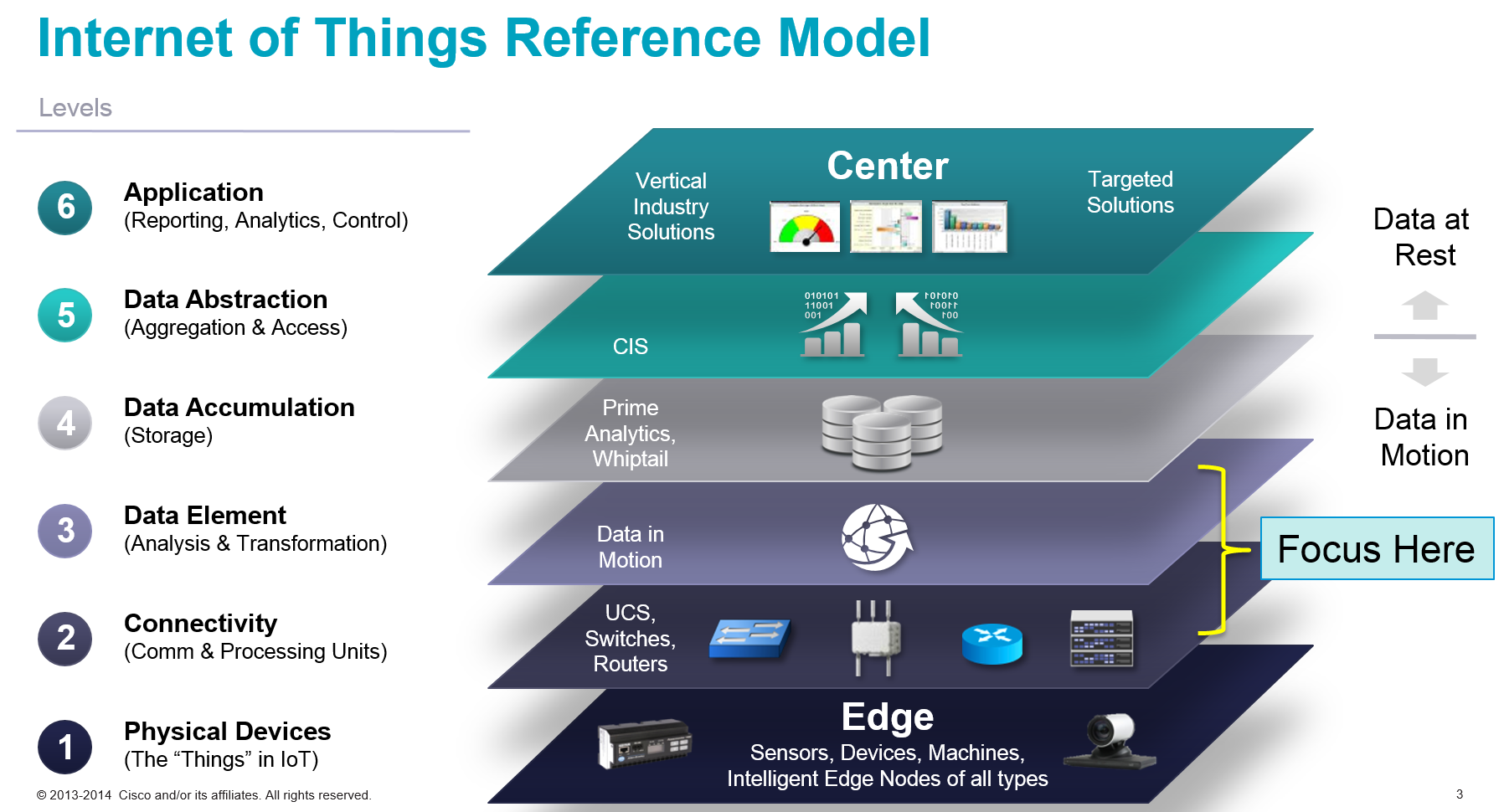
We are going to use the M2M structure in the project. Here is a top level introduction of M2M:( the structure introduction comes from TS-0001-V1.5.0--2015-January-23--OneM2M Technical Specification)



1）Application Entity (AE): Application Entity is an entity in the application layer that implements an M2M application service logic. Each application service logic can be resident in a number of M2M nodes and/or more than once on a single M2M node.

2）Common Services Entity (CSE): A Common Services Entity represents an instantiation of a set of "common service functions" of the M2M environments. Such service functions are exposed to other entities through the Mca and Mcc reference points.

3）Underlying Network Services Entity (NSE): A Network Services Entity provides services from the underlying network to the CSEs.

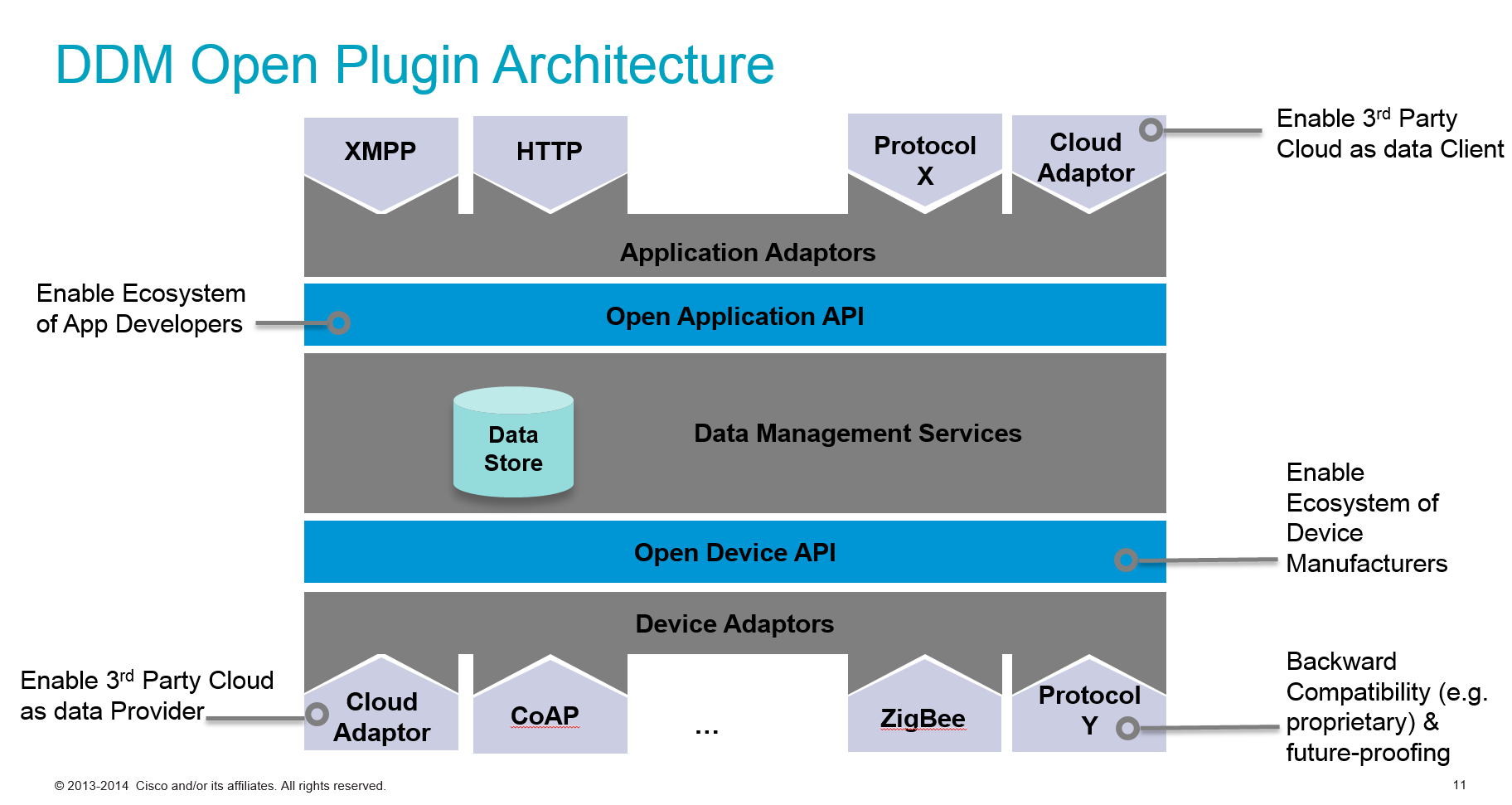


In this project, we are not going to focus on a device-level API, instead we will develop in a database-level API with the assumption that all the data come from a database.

The core part of the project lies in the optimization of the visualization. How to create a satisfying response time while maintaining a Show-What-The-User-Wants manner. Several brilliant proposals have been created and we need a deeper discussion with the mentors to figure out which method is applicable to the project.

A step further will be how to organize and analyze the data which plays a bonus role in our project.

The SDN we use will be based on OpenDayLight(ODL). Here, we are not going to develop through the ODL API or function but using the ODL SDN function (e.g. create nodes, configure the topology, attach the interface etc.) in general.



We are going to mainly focus on the southbound protocol in this part while getting familiar with the northbound protocol. Here we will use HTTP, XMPP and other existing application level protocols and we will try to integrate them into the same interface.

Within the northbound protocol, CISCO has done great job on this and we will follow the documentation, building a good knowledge of both the conceptual and brief implementation of the protocols.

The data storage strategy is another critical topic we are going to focus on. In this topic, we will develop a fast and reasonable way to fetch and update data(e.g. we may use the node cache to accelerate the response speed)

Key challenge here is a good understanding of various protocols and how to apply them to the system. To begin with, we may use the basic HTTP protocol.

The tool we are going to use:

D3(JQuery): Using this library to render the tree structure of the layer and data retrieved from the database

MongoDB: The database used to store the IoT data

Model–View–Controller(MVC): A software design pattern that we are going to implement into the UI design.

Design Implications and Discussion:

How to make the tree structure visualized more appropriately: With 1 million + nodes and bunch of layers implemented in the database, we decided to use different kinds of graph to render the relationship between each layer and node, for example using the bar chart to let the user know the distribution of the data in each container, using the collapse tree to render the specific node (actual data) in the container. All of these graphs should have some connection between each other and give a clear view of the relationship between the level, for example there are five high-level containers that keep different types of data (temperature, humidity, etc.), each high-level container has 50,000 containers (sensor devices) which have X nodes (points of data), this three level relationship should be rendered clearly through the graph.

How to sort the data out: When the user are not searching for specific data in the database, there should be some elements that can be sorted as a keyword, such as time/location/data\_type. For example, we can start with year, and each layer farther outward apply year -> month -> day -> AM/PM -> hour -> minute ect. With the location, we can group the data in layers from continent -> country -> state -> city. For data\_type we can have alphabetical ranges in each layer to narrow down the number of data types. For example to find temperature select S-Z in the first layer, and then if # of data\_types is still too large select letter T. Then select temperature from the data\_types starting with letter T.

**4. Acceptance Criteria**

Minimum acceptance criteria for this project is information visualization of 1M+ nodes. The stretch goals are optimizing the client-side response time in the web browser and implementing management functions for adding or deleting nodes from the tree.

**5. Release Planning**

release #1 (Due by Week 6)

First demo on the project.

release #2

release #3

release #4

\*need to talk with client Tuesday evening to get a better understanding of the technical implementation pieces to be able to allocate release planning for demo-days. Will update for final verison.