1 Overview

1.1 Modern Wireless Connectivity

Modern Wireless Devices are self-serving. A smart phone can be connected to a 4g network, a 3g network, an LTE network, a Wi-Max network, a WiFi network, or any other type of network for which the device contains an appropriate radio receiver. The device itself connects to the network chosen by the user. This is generally an easy choice for the user; if the 4g network has a low signal strength, it is easy to switch to a local wireless network if one is available. This seems effective from the perspective of the device owner, but having every wireless device owner select their own network of preference may not be an optimal setup.

1.2 A Proprosal

A better approach may be to have a device controlling the network distribution in a local area. A device called a Global Resource Controller (GRC) could receive data about devices and network traffic and make a decision about which network each individual device should use. This requires what we will call a Heterogenous Network (HetNet). HetNets are a composition of all the available wireless networks in an area; in our model, a device could make a seamless transition between these networks at the behest of the GRC. The GRC's job is to ensure that thee devices in a HetNet are allocated to networks in an optimal manner. Optimality could depend on a number of variables; our goal could be to maximize bandwidth, cost efficiency, or a number of other variables. Our goal is to design a tool that can simulate this kind of environment to allow students a more visual experience when learning this concept, as well as allowing research to be done into the feasibility of this kind of setup.

1.3 Our Tool

We are in the process of designing a web tool that allows a user to design a visual model of a HetNet. We believe that these types of optimality problems can be represented as Linear Programs and then solved. Our HetNet simulation should be capable of taking a given HetNet, optimizing

it, and returning the results in a visual fashion. In the next section, we will discuss the design and implementation of this tool.

2 Tool Design

2.1 Overview

As seen above, our web tool uses a typical web client-server model. The web client will expose a GUI to the user, allowing them to build a visual model of a HetNet. Once the user is satisfied with their model, it will be submitted to the server. The server will request a solution to a Linear Program using the information from the HetNet model. This Linear Program will then be submitted to our Linear Program Solver. If all goes well, a solution will be generated and returned to the client in some visual fashion. We are in the process of creating an error handling system in the event that problems generated are infeasible or degenerate.

2.2 Implementation

The client will be created using JavaScript libraries for creating the visual elements and a JSON message passing system for communication to the server. The server will be running a PHP backend which will invoke optimized C++ code containing the Solver logic. The logic is an implementation of the two-phase simplex method. We have used the tool *SWIG* to generate PHP to C++ bindings.

2.3 Solver Design

The Simplex Solver has been designed in a typical object oriented fashion. It specifies a format for Linear Programs which is exposed by the Linear Program class. Linear Programs can be created and constraints can be added before the Linear Program is passed to the Solver. The Solver is an implementation of the Singleton design pattern, allowing for a single, lightweight, lazily generated object to handle any number of threaded accesses. This is with the intention of adding support for "dynamic models"; models that will track devices in the HetNet over a specific timeframe; this could only be done in a timely manner using multiple threads to generate the response.