Lit Banjo; Keen Hoes: a 6.033 Design Project

Obasi Onuoha, Kenneth Friedman, Joel Gustafson

Abstract—Improving the performance of the wireless network is of obvious interest to MIT. The task is to implement a system which optimizes the connections between clients and the MIT wireless network in order to improve the overall performance of the network. This paper proposes such a system.

The design revolves around three modules: each client will have a *controller* which communicates its requirements to an *access point* which collects data about the traffics and clients to send to the *server*. The *server* ingests data about the network, stores it, and distributes connection instructions to the rest of the network.

1 High Level System Overview

Our project increases net awesomeness by 2000%.

1.1 Subsection Heading Here

Subsection text here.

1.1.1 Subsubsection Heading Here

Subsubsection text here.

2 Client

Are the machines the clients or are we?

3 Access Points

Networks can have tens of thousands of access points (APs), and may install, move, or remove them frequently. In order to keep all the access points easily maintainable, our system design uses AP primarily as thin clients to forward data and requests to and from the central server and the client, simplifying the design and minimizing AP logic. This section outlines the functions of each AP beyond serving client requests for data from the outside internet, while section 5 provides a technical specification for communication between network devices.

3.1 Client Connection

When a client connects to an AP, the AP forwards the client's initial connection message to the server, as detailed in sections 5.1 and 5.3. At this point, both the AP and the client assume that the client will remain connected to the AP unless explicitly reassigned by the server - a response is not needed or expected. This both reduces the number of messages and preserves a moderately functional network in event of a server failure.

3.2 Client Reassignment

Upon receiving a client reassignment message, an AP will forward it to the appropriate client, if it exists. The AP assumes no responsibility for ensuring that the client disconnects, and will continue serving its requests in the interim. The format of these messages are detailed in sections 5.2 and 5.4.

3.3 Network Statistics

Every 30 seconds we send stuff.

3.4 Client Dissatisfaction

Say no to drugs, kids.

4 Server

The majority of the computational complexity is handled by a central server, connected via reliable wired connection to all APs. The server aggregates network data and handles client assignment requests from APs. This section details the components of the central server and their interactions with the rest of the system.

4.1 Network State

The server maintains the state of traffic in network in a 6.006 data structure. Every 30 seconds, God rips the wings off of an angel, and Donald Trump sacrifices a campaign manager to Ba'al the Soul-Eater.

4.2 Client Assignment

The primary role of the sever is to respond to client assignment requests from APs with the MAC address of the best AP for the client to connect to. This routing function is computed by a neural net.

4.3 Network Analytics

In addition to client assignment routing, the server tracks network traffic statistics and reports aggregated data to IS&T.

5 COMMUNICATIONS PROTOCOLS

This section outlines the communications protocols used from the controller of a client to the AP, from the AP to the controller of a client, from the AP to the server, and from the server to the AP.

5.1 Client to AP

When a client connects to an AP, its controller immediately send a frame to the AP. The frame is of the following form:

src addr |*dst addr* |*meta* |*data*

Where *src addr* is the 48-bit MAC address of the client, *dst addr* is the 48-bit MAC address of the AP to which the controller is communicating, and *meta* is the 8-bit value 00000001. *Data* is a variable-bit value defined as follows:

R |*addrs*

Where *R* is a 32-bit integer representing the maximum number of bits that the client will need to transmit over the course of any one second and *addrs* is the value formed of the concatenation of the 48-bit MAC addresses of all of the APs within range of the client. The MAC addresses which compose *addrs* are sorted in decreasing order of signal strength.

Every 30 seconds, the client sends a message to its AP. This message is of the following form:

src addr |*dst addr* |*meta* |*data*

Where *src addr* is the 48-bit MAC address of the client, *dst addr* is the 48-bit MAC address of the AP to which the controller is communicating, and *meta* is the 8-bit value 00000100. *Data* is a variable-bit value defined as follows:

G |*A*

Where *G* is a 32-bit integer representing the number of bits that the client generated over the past 30 seconds and *A* is a 32-bit integer representing the number of bits that the client successfully sent over the past 30 seconds.

5.2 AP to Client

When an AP needs to tell a client to connect to a different AP within range of the client, it sends a frame of the following form:

src addr |*dst addr* |*meta* |*data*

Where *src addr* is the 48-bit MAC address of the AP sending the frame, *dst addr* is the 48-bit MAC address of the client to which the AP wished to communicate, and *meta* is the 8-bit value 00000010. *Data* is 48-bit value specifying the AP to which the client should connect.

When an AP needs to tell the user of a client to move physically in order to connect to a different AP which is not in the immediate range of the client, it sends a frame of the following form:

src addr |*dst addr* |*meta* |*data*

Where *src addr* is the 48-bit MAC address of the AP sending the frame, *dst addr* is the 48-bit MAC address of the client to which the AP wished to communicate, and *meta* is the 8-bit value 00000011. *Data* is a 24-bit value defined as follows:

bld |*rm*

Where *bld* is a 12-bit binary integer specifying the building number of the desired AP and *rm* is a 12-bit integer specifying the room number of the desired AP.

5.3 AP to Server

When a new client connects to an AP, it sends a message to the IS&T server. This message is of the following form:

maddr |*caddr* |*R*

Where *maddr* is the 48-bit MAC address of the AP, *cddr* is the 48-bit MAC address of the client which just connected, and *R* is a 32-bit integer specifying the maximum number of bits that the client will need to transmit over the course of any one second.

Every 30 seconds, independent of any connected clients, the AP sends a message to the IS&T server. This message is of the following form:

maddr |*cnum* |*rsum* |*asum* |*gsum*

Where *maddr* is the 48-bit MAC address of the AP sending the message, *cnum* is a 7-bit integer specifying number of clients connected to the AP sending the message, *rsum* is a 39-bit integer specifying the maximum number of bits that the clients connected to the AP sending the message may need to send over any given second, *asum* is a 20-bit integer specifying how many bits clients have transmitted to the AP sending the message over the last 30 seconds, and *gsum* is a 39-bit integer specifying the number of bits that the clients connected to the AP sending the message have generated over the past 30 seconds.

5.4 Server to AP

When the IS&T server determines that an a client needs to connect to a different AP, it sends a message to the AP that client is currently connected to. This message takes the following form:

caddr |*naddr* |*rlct*

Where *caddr* is the 48-bit MAC address of the client which is being directed to switch to a new AP and *naddr* is the 48-bit MAC address of the AP to which the client is being directed to switch. *rlct* is a 24-bit value composed of all 0's if the AP in question is in range of the client in question or a 24-bit value defined as follows if it is not:

bld |*rm*

Where *bld* is a 12-bit binary integer specifying the building number of the desired AP and *rm* is a 12-bit integer specifying the room number of the desired AP.

6 CONCLUSION

We have presented this proposal in order to improve the quality of the MIT wireless network. We designed the system to be extremely scalable. Utilizing a neural net

allows the system to constantly update its assignment priorities as the network and its needs grow and change. Additionally, the adoption of this architecture realizes a very high performance system. Constant and consistent optimization ensures that desirable outcomes will be regularly attained. Finally, our system places the primary load of computation on the server. This greatly simplifies the system and means that the majority of changes will not need to be propagated through the network.

Future work will require us to train the neural net and to evaluate the performance gains.

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.