Fast-Fi: Designing Large Throughput-Optimized Wireless Networks

Brandon Carter[1], Geronimo Mirano[1], Helen Zhou[1]

Abstract

Large wireless networks often have difficulty handling situations in which many clients in close proximity join the network in a short time. Performance can also degrade when serving clients that have different usage patterns. We propose Fast-Fi, a system that seeks to maximize throughput for clients and provide a better network experience. Fast-Fi employs traffic control mechanisms on both the client and access point (AP) machines, helping clients to choose APs such that their usage is distributed optimally across in-range APs. Additionally, it suggests nearby networks for users if no networks in-range can provide sufficient throughput. Moreover, the system collects usage patterns from all access points into a centralized server database for network administrators. This system is both distributed and scalable such that it can be used in networks even larger than that at MIT. We show that this model improves upon the current MIT network in a variety of common usage scenarios.

[1] *Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA*
{bcarter, geronm, hlzhou} @mit.edu

Contents

1. Introduction

Wireless networks aim to provide users a fast, convenient way to connect to the internet over a large area. These networks consist of access points (APs) which communicate with client devices that are within range. Data flows from clients through the wireless network and then out to the rest of the internet. Large wireless networks, such as that at MIT, consist of thousands of access points and tens of thousands of clients. Because a device might be in range of multiple access points, the network and device must choose a suitable AP that will yield optimal performance. Moreover, because some clients might consume more network bandwidth than others, the system must also ensure that network performance does not degrade for other users. However, current wireless networks often have difficulty achieving these goals.

We propose a system, Fast-Fi, which addresses this problem and seeks to provide clients improved performance and maximal throughput on a large wireless network, even under periods of heavy network load. The system secondarily maintains historical data and statistics on connected devices to aid network administration. The system design facilitates modularity and scalability.

2. System Modules

2.1 High Level Overview

Our goal is to provide high-utilization internet coverage to a large campus area. In our model, numerous APs are placed around this campus, and users' client devices connect to these APs for internet access. The APs can communicate wirelessly with clients and over the wired network with each other, subject to range limitations. Many APs will often be within range of a single client, thus allowing for some flexibility in how users connect. These APs are connected to a wired network which they use to serve internet requests. They are also connected to a central IS&T (Information Services and Technology) server, which can be used for data logging and

for retrieving information about other APs in the campus network.

2.2 Client

Clients are devices such as laptops and phones that send information over the Internet by means of APs on the MIT network. Clients can have very different usage patterns. Some clients require a large number of bytes per second, for example while streaming live video, while others have much less stringent network requirements, for example when reading email.

Each client device is uniquely identified by a MAC address and is equipped with two pre-defined software modules. The *monitor* tracks running applications and network performance, computing throughput for the devices, while the *controller* communicates with APs and determines which one to connect to. We modify the controller to communicate according to the protocol discussed in Section 4.

2.3 Access Point

Every access point (AP) belongs to the campus network and is uniquely identified by a MAC address. As specified, up to 128 clients can be simultaneously connected to a single AP. The role of the AP, in addition to serving its connected clients, is to maintain various status information about nearby APs and transmit this data to client devices. APs also collect their own usage data and send dumps of this data to the IS&T server.

To maintain awareness of nearby APs, each AP sends a request to the IS&T server once every six hours to obtain information about its neighbors. In response, the AP receives a list of AP addresses sorted by geographic distance. The AP, then, is able to communicate with these APs over the wired network. By enabling this direct communication within clusters of APs, our design pushes the work to APs distributed throughout the network.

2.4 Central IS&T Server

The central IS&T server has three functions: first, it knows and can serve the MAC address and location (including building/room information and geographical coordinates) of every AP in the network; second, it provides the ability for the APs to learn information about nearby APs on the network; lastly, it stores usage history from APs for network administrators.

To register new APs into the network, the central server will be manually updated with their addresses and locations. The update is done manually in order to prevent unintended registration of malicious APs. Every six hours, the central server receives a request from each AP to learn about nearby APs within a distance D. The server then computes the geographic distance to all other APs (using the building/room and geographical data), and returns the nearby APs in a sorted list.

To store network usage information, the IS&T server maintains a SQL database, keying data using a unique ID for each of the APs. The server maintains an interface for receiving periodic data dumps from the APs (described in detail in 3.5) and then runs a script to import these dumps into the database. The network administrators can easily run queries on and export data from this database at any time.

3. System Design

3.1 Address Setup and Update

When the system is first booted, each of the APs in the network must learn the addresses of neighboring APs so that they can be contacted via wired network. We discuss an address-updating scheme, the first iteration of which sets up the network such that all APs have the addresses they need.

After the initial setup, each AP updates its set of addresses of nearby APs by contacting the IS&T server every six hours. An AP queries the IS&T server with a distance parameter D, which specifies the radius of neighboring APs it wishes to learn the addresses of. For our current specifications, $D = 625$ ft is a good value, though this value may be adjusted as the strength of APs vary or the client wishes to know about further APs when nearby APs are unsatisfactory. The IS&T server, which maintains a map of all of the APs and their corresponding geographic locations, returns all APs within distance D of the AP that sent the request (in order of increasing distance). This address-updating system is depicted in Figure 1.

In Figure 1, we see AP1 contact the the IS&T server for information about its surrounding APs. The server returns the addresses of APs 1 through 4, which AP1 can now redirect clients to.

Note that in a network with n APs, when the network is first set up, $\Theta(n^2)$ distances must be calculated and cached (the distance between all pairs of APs). Sorting the distance from each AP to all other APs is done naively in $\Theta(n^2 \log n)$ time, which is quick for $n = 4000$ APs. Given that this computation only needs to be done once, and all future additions of APs can be done in $O(n^2)$ time (though expected $\Theta(n \log n)$ time, since the AP graph is not dense), the overhead incurred is rather small and very worthwhile. If n were sufficiently large, these update algorithms could be improved, though they would incur a larger overhead, which is why we opt not to use them for now. This implementation for updating should allow these computations to be done quickly relative to the latency of the wired network.

3.2 AP Selection for Maximal Client Satisfaction

Our system seeks to maximize client satisfaction, or *happiness*. Happiness is operationally defined by the degree to which a user has indicated unhappiness in the client software. In the context of AP selection, we assume that clients will be happiest when their throughput is maximized.

In particular, our system maximizes each user's predicted network throughput. Each user's client controller chooses an in-range AP with a suitably high average predicted throughput for the next few seconds, based on information about APs' current and past usages.

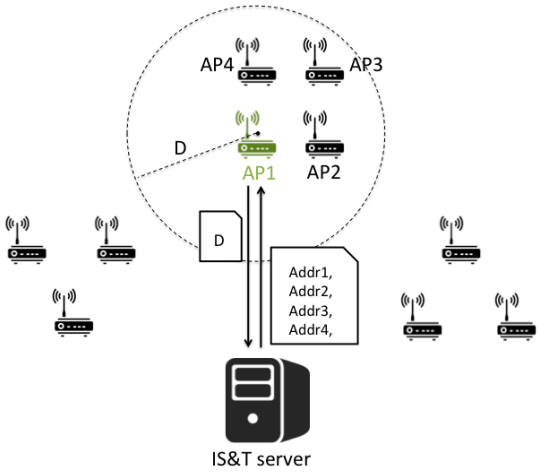

Figure 1. Fast-Fi's address-updating scheme. Every six hours, each AP requests the updated addresses of surrounding APs within physical distance D of itself.

3.2.1 AP Selection

In our system, all APs maintain a list of nearby APs' locations and IP addresses, as explained in 3.1. APs can use this list to query other APs for their usage and performance using the wired communication link.

Given that the APs have information about nearby APs, it is relatively straightforward for the client to learn which of the APs in its range can accommodate its traffic (which is at most a known value of R bits per second). When first attempting to connect, the client picks one in-range AP and asks it for usage information on all the other APs that the client sees. This first AP will query the other APs in IP-closeness-order (starting with itself) until it finds one with throughput satisfying the user's criteria, at which point it will return this recommended AP. If no in-range APs have sufficient throughput available, then the system offers the maximum-throughput AP to the client, and also suggests an out-of-range AP (further detailed in Section 3.3). The client is thus able to discover information about many of the surrounding APs without having to connect to them individually.

3.2.2 Throughput Prediction

Next, we specify how expected throughput is calculated. A client's throughput depends on (1) the number of other connected clients c, (2) the current total throughput for these clients T, and (3) the AP's channel capacity $T_{\max}$. Because APs can be highly volatile, rather than using the current values of these parameters, we perform a regression to produce the functions $c(t)$ and $T(t)$. With this information, we can determine the average projected throughput for the next τ seconds as:

$$T_{\text{proj}}(\tau) = \frac{\int_0^\tau \max(T_{\max} - T(t), \frac{T_{\max}}{c(t)+1})dt}{\tau} \qquad (1)$$

for a suitably chosen value of τ. This should allow maximization of expected user throughput. We will use simulation to evaluate various regression kernels to capture the information contained in $T(t)$ and $c(t)$. Among the simplest such regressions would be a linear regression, but we conjecture that a k^{th}-order polynomial regression or a multiple exponential regression will produce better approximations. In this way, the system will take into account the momentum inherent to volatile AP connections when deciding which APs will provide optimal throughput. The value $T_{\text{proj}}(\tau)$ can be compared against R, the client's maximum desired throughput (which the controller knows). If an AP can be found satisfying the above, the client's throughput should be well-accommodated. As described above, if no in-range APs are satisfactory, the client is connected to that with the greatest predicted throughput $T_{\text{proj}}(\tau)$. This procedure gives clients the best performance possible under most network conditions.

3.3 Out-of-Range AP Recommendation

When all APs in range of the client are considered *unsatisfactory*, the AP that the client is connected to should recommend another AP within 500 ft of the client.

There are two cases under which the in-range AP's are unsatisfactory: (1) when they all have 128 clients currently connected to them, or (2) when they all have less than R available throughput (where R is the client's maximum required throughput, as defined in Section 3.2).

Because the average range of an AP is 125 ft, the client may not be able to detect all APs that are further than that but are within 500 ft of itself. These APs' addresses are instead known by the AP that the client is currently connected to (call this AP_c). It knows this because during the setup phase of Fast-Fi all APs are equipped with the addresses of all other APs within $D = 625$ ft of themselves, as described in Section 3.1. AP_c then asks for the projected throughputs of of each of these APs, until it finds one with sufficient room for the client's R bits per second. The user is then given a suggestion to relocate to join this satisfactory AP. Figure 2 illustrates this situation.

3.4 Monitoring Throughput and Happiness

The monitor on the client keeps track of the device's throughput. Since the network conditions on each AP fluctuate frequently, the system considers other available APs and either switches the client to a more optimal AP when the client reports dissatisfaction or suggests a better out-of-range AP. Finding a better AP is computed using the same procedure to maximize happiness as if the client were connecting initially (Section 3.2), and suggestion is done using the procedure outlined in Section 3.3. We define a client to be dissatisfied: (1) when $A < G$, where G is the client's bits generated by applications and A is the client's actual sent bits, or (2) when the user clicks the "unhappy" button through the user interface on the client. Once the *dissatisfied* state is set, it is only unset once $A = G$, that is, when the client again gets its desired throughput.

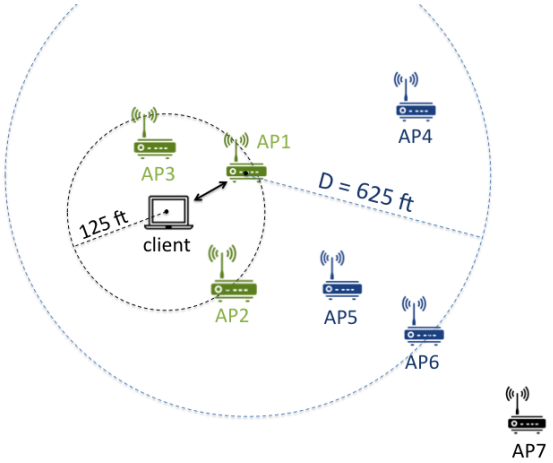

Figure 2. When a client's in-range APs (APs 1-3) are unsatisfactory, the AP that the client is connected to (AP1) will look through the addresses of other APs that it knows to be within 625 ft (APs 4-6) and suggest satisfactory APs to the client.

3.5 Usage History Logging

Every second, each AP appends current usage data to a running binary dump. Each entry in the file contains the current UNIX timestamp (32 bits), the number of bytes sent through the AP in the last second (by taking the current number of bytes from the running 32-bit counter), and the number of users currently connected (as a 12-bit number). Every 24 hours, the AP flushes the binary dump to the IS&T server, which parses the data into a SQL database.

This binary storage format is both efficient and scalable. The maximum file size each day is roughly 6.5MB. The format will work to store AP throughputs up to 34Gbit/s and 4096 connected users and is easily generalizable if these numbers grew larger.

4. Communication Protocols

The APs and client controllers that transmit messages use the following protocol. As shown in Figure 3, each packet contains a 48-bit source, a 48-bit destination, an 8-bit meta value, and a d-bit data block. The meta values indicate the type of communication. The possible meta values in our communication protocol are enumerated in Table 1.

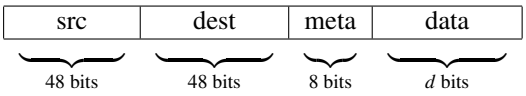

Figure 3. Structure of a packet

5. Use Cases

The following situations explain how this system operates under a variety of common use cases.

Table 1. Table of Meta Values

Meta Value	Description
00000000	*Application Data* Client → AP: outgoing packet data AP → Client: incoming packet data
00000001	*Clients getting a satisfactory AP from among in-range APs* Client → AP: list of in-range APs' IDs; R; τ AP → Client: ID of a single recommended AP; $T_{proj}(\tau)$
00000010	*Clients getting satisfactory APs that are nearby but out of range* Client → AP: R; τ; num results to return k; max distance to AP D AP → Client: list of IDs, throughputs $T_{proj}(\tau)$, locations of k recommended AP
00000011	*APs getting other APs' throughputs* AP1 → AP2: τ AP2 → AP1: AP2's current throughput $T_{proj}(\tau)$
00000100	*APs getting nearby APs* AP → IS&T: distance D (in feet) IS&T → AP: list of IDs, locations, IP addresses of sorted APs within D feet from the requesting AP

5.1 Single Client, Network Underutilized

When a solitary client goes through the AP selection process, it sees all of the APs that are in-range and connects to one of them to find one which is *satisfactory* (as defined in Section 3.3). Because this AP itself is currently underutilized, it will suggest itself as the best connection point and immediately begin serving the client's traffic, making for a very fast time-to-connect. Regardless of whether the client is a small client or a large client, the AP ought to be able to handle all of the client's traffic, as there is little competing throughput. Thus, the system operates well under this condition.

5.2 Many Clients, Single Room

Consider the case where hundreds of clients enter into a room with multiple APs. Because each client considers more-crowded APs less desirable, clients will naturally distribute their connections evenly among the APs. How things proceed beyond that depends on certain features of the situation. If all clients are in range of all APs, and these APs can together can accommodate all the clients' traffic, then the above load-balancing will ensure full accommodation. On the other hand, it might be the case that some clients are in range of only a fraction of the APs. In this case, these APs are liable to become throughput-limited, putting their clients' controllers in an *dissatisfied* state. At this point, those clients which are able to see freer in-range APs will switch, thus decreasing the load on this fraction of APs. This will improve the performance for the users remaining on these APs, and thus good utilization is still achieved. In the most extreme case, where there isn't enough bandwidth to accommodate all the clients, then full utilization will be achieved as connecting clients

attempt to maximize their throughput; and meanwhile, clients will each be recommended the location of another satisfactory AP within 500 ft.

5.3 Moving Clients

In regions of high churn, the network will experience frequent connections and disconnections. Because our system requires APs to cache the throughput information for their neighboring APs and only update this information after a set interval of time, the workload the Fast-Fi places on the APs should remain manageable. Furthermore, since Fast-Fi connects the client to a satisfactory AP as soon as one is found, little computation has to be done with each connection. In a later iteration of our system, we believe that analyzing the movement of students could be beneficial for less transient connections. That is, if a student is walking down a hallway, we could predict the next AP the student will pass, and prioritize it more when choosing from possible APs to connect to.

5.4 Scaling Beyond MIT

The proposed system is robust to changes in scale, and does not rely on the structure of MIT's network.

Due to the system's mostly distributed nature, new APs can easily be added to the system. A new AP simply contacts the central IS&T server, and the IS&T server sends back the addresses of the APs within $D = 625$ ft of the new AP. The IS&T server also updates those APs within $D = 625$ ft of the new AP with the address of the new AP. We note that the size of these updates is $2 * (n_D - 1) = \Theta(n_D)$, where n_D is the number of APs packed in a circle of radius D around the new AP. This is because IS&T sends $n_D - 1$ addresses to the new AP, and sends the address of the new AP to each of the $n_D - 1$ surrounding APs. Thus, the system scales in address updates with n_D. Since the density of APs that can fit within $D = 625$ ft is practically limited, this aspect of Fast-Fi scales very well.

An increase in the number of APs would also result in more logs being sent to the IS&T central server. The information being sent to the IS&T server is relatively small and compressed but needed to facilitate network analysis (bytes transferred and number of clients connected). Depending on how many APs are in the network and how often the network administrators flush the log data, the IS&T server may eventually run out of storage. This is unavoidable given the need for log information to be sent to the IS&T server, so administrators should set periodic flushes of stored logs once the information becomes obsolete or arrange for a distributed database across multiple server machines.

Overall, Fast-Fi's approach of pushing the work out to the APs minimizes the load on the central server, and lends itself very well to scaling to large networks.

6. Conclusion

We propose a large-scale wireless network in which clients connect to access points that provide optimized connections for their specific needs. APs obtain information about nearby APs such that clients looking to connect can find an AP which can provide sufficient throughput. If all in-range APs are unsatisfactory, the client can request information on other APs in close geographical proximity that would provide greater bandwidth. If network conditions degrade or the client becomes dissatisfied after joining an AP, the system will attempt to switch the client to a better network or suggest alternative nearby APs.

This mechanism is scalable for large networks and performs well in the context of common usage scenarios, providing clients with optimal network experiences. Moreover, this system monitors network usage at each AP and stores historical data on the IS&T server.

Future design goals include allowing the users more control over the specific parameters used in network selection, as well as improving the mechanisms for shuffling users between APs as network conditions degrade. As it stands, the system provides a scalable, modular solution as required by the network's users.