MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 DESIGN PROJECT

Fast-Fi: Designing Large Throughput-Optimized Wireless Networks

Brandon Carter, Geronimo Mirano, Helen Zhou
{*bcarter, geronm, hlzhou*}*@mit.edu*

Matei Zaharia
Cathie Yun
R01/R09 TR10/11

May 6, 2016

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 Fast-Fi, a system 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, scalability, and fault-tolerance.

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 with each other over the wired network and with clients wirelessly, subject to range limitations. The wired network allows the

APs to serve data from the internet to clients. Many APs will often be within range of a single client, thus allowing for some flexibility in how users connect.

The APs are also connected via the wired network to a central IS&T (Information Services and Technology) server, which is used for data logging and for managing 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, whereas others have much less demanding 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. 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. Six hours provides a balance between refreshing information fairly often and not over-utilizing network and machine resources. In response, the AP receives a list of AP addresses sorted by geographic distance. The AP, then, is able to communicate with these nearby APs over the wired network. By enabling this direct communication within clusters of APs, our design pushes the work of updating status information 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 is 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 list sorted by distance.

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 Section 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.

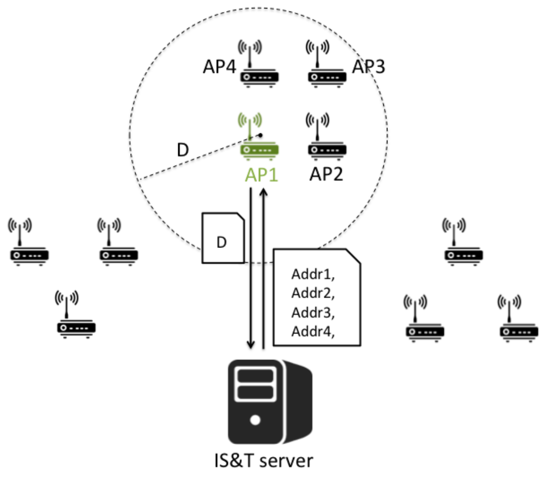

Figure 1. Fast-Fi's address-updating scheme. Every six hours, each AP requests the updated addresses of surrounding APs within physical distance D of itself.

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 sparse), the overhead incurred is rather small and very worthwhile. If n were sufficiently large, these update algorithms could be improved by using more sophisticated data structures. However, they would then incur a larger performance 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 In-Range AP Selection

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 or to which the user does not attain desired throughput. 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.

3.2.1 AP Selection

In our system, all APs maintain a list of nearby APs' locations and IP addresses, as explained in Section 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 other nearby APs, it is easy for the client to learn which APs can accommodate its traffic (at most a known value R bits per second). The client begins by cycling through every WiFi channel to discover in-range APs. After first seeing any Fast-Fi AP, the client spends t_{scan} seconds searching for yet more APs, so that it has some additional knowledge about what APs are in range and can choose the best one available. After t_{scan} seconds, when first attempting to connect, the client picks one in-range AP (at random) and asks it for usage information on all the other APs that are in range of the client. This first AP will query the other in-range APs in IP-closeness-order (to minimize network usage) starting with itself until it finds one with throughput satisfying the user's criteria, at which point it will return this recommended AP.

These requests are cached with an expiry time of 1 second, so if an AP needs to provide information about a nearby AP that it has requested information from within the past second, it uses the cached version of the request to save network resources. 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 expected throughput on a given AP 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 on the number of connected clients 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:

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

for a suitably chosen value of τ. This computation should allow maximization of expected user throughput. By evaluating the errors after fitting according to various functions (exponential, logistic, linear, and low-degree polynomials), we capture the information contained in $T(t)$ and $c(t)$.

Among the simplest such regressions would be a linear regression over data for the number of connected clients, collected at a frequency of once per second, stored in a queue (for efficient push/pop operations) containing a history of the most recent 1-2 minutes. We thereby capture short-term trends without requiring the storage of too much data.

While a linear regression with a 1-2 minute history may provide a sufficient and efficient approximation, we conjecture that a higher-order polynomial regression, logistic regression, or an exponential regression may produce better approximations over slightly longer time periods such as 5-15 minutes (discussed in Section 5.3). 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 client's throughput requirement, the client's throughput should be well-accommodated. Otherwise, as described above, if no in-range APs are satisfactory, the client is connected to the AP 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).

Since the average range of an AP is 125 ft, the client may not be able to detect all APs that are further than 125 ft but are within 500 ft of itself. These APs' addresses are instead known by the AP that the client is currently connected to (which we denote AP_c). AP_c has this information because during Fast-Fi system initialization, all APs are provided 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 (in increasing order of distance), until it finds one with sufficient room for the client's R bits per second. Because the IS&T server provides each AP with a list of other nearby APs sorted by distance, this procedure simply requires iterating through the sorted list of nearby APs and requesting their throughput information. The user is then given a suggestion to relocate to join this satisfactory AP. Figure 2 illustrates this situation.

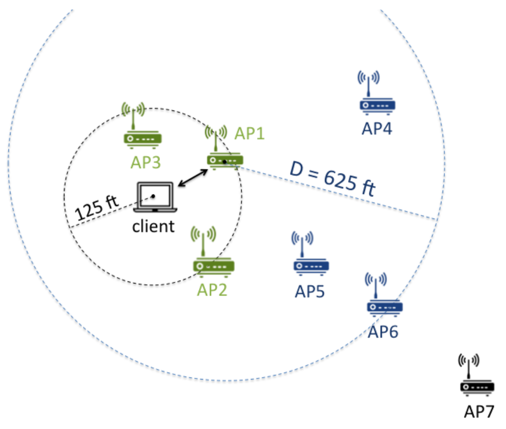

Figure 2. When all of a client's in-range APs (1-3) are unsatisfactory, the AP that the client is connected to (1) will look through the addresses of other APs it knows to be within 625 ft (4-6) and suggest satisfactory APs.

3.4 Monitoring Throughput and Happiness

Next, we will describe how our system keeps clients happy during use. We define a client to be *dissatisfied*: (1) when $A < G$, where A is the client's actual sent bits and G is the client's bits generated by applications, 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.

As long as a client is dissatisfied, the system considers other available APs and either switches the client to a better AP or suggests a better out-of-range AP. The automatic switching procedure comes with a cooldown timer, to insure that clients are not switching APs too frequently. The mechanism for discovering in-range APs is the same as when establishing an initial connection (Section 3.2), and out-of-range AP suggestion is done using the procedure outlined in Section 3.3.

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 in the AP), 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. These log files are transferred from APs to the server using SFTP.

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.

$$\underbrace{\boxed{\text{src}}}_{48\text{ bits}}\underbrace{\boxed{\text{dest}}}_{48\text{ bits}}\underbrace{\boxed{\text{meta}}}_{8\text{ bits}}\underbrace{\boxed{\text{data}}}_{d\text{ bits}}$$

Figure 3. Structure of a packet

The possible meta values in our communication protocol are enumerated in Table 1. The table makes reference to a client's desired throughput R, the number of seconds we want to project into the future τ, the projected throughput for an AP $T_{proj}(\tau)$, and the maximum acceptable distance that a client is willing to move D.

4.1 IS&T Server Downtime

In communications involving the IS&T central server, there can be instances where the server goes offline and is unreachable for a up to two minutes. Our system works around this issue. If a request from an AP to the IS&T server times out, then the AP knows to cache the request and retry it again in two minutes, when the server will be back online.

The system as a whole does not degrade in performance or user happiness in the case of IS&T server outages. The design attempts to be fault-tolerant with regard to downtime on the IS&T server. Instances where APs contact the IS&T server include fetching information about nearby APs and uploading usage history logs. We assess fault-tolerance for both of these communications.

In the event that the IS&T server does not respond to the nearby AP information request, the only downside is that an AP would not be informed of any other APs that have joined or been removed from the network since it cannot request

Table 1. Table of Meta Values used by Fast-Fi

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 requesting other APs' throughputs* AP1 → AP2: τ
00000100	*APs sending their throughputs to other APs (in response to 00000011)* AP2 → AP1: AP2's current throughput $T_{proj}(\tau)$
00000101	*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

new contact information about nearby APs. The AP would continue to have in its cache the IP addresses of the previous nearby APs, so it would continue requesting up-to-date status information of these nearby APs, regardless of the status of the IS&T server.

To address the second concern of the usage history log needing to be maintained on the AP for an extra two minutes, there is ample storage to begin the next day's log with two minutes worth of information. The only issue then is that the usage history data stored in the database might be two minutes old at any time. Though, since the server administrators are aggregating usage data over a large period of time, this small lag is not a problem (and in fact, the administrators would be unable to query the database while the server is offline anyway).

Therefore, since new APs are added to the network infrequently and there is no problem storing the log on the AP for two extra minutes prior to sending it, the effects of the two minute downtime on the IS&T server are negligible to none in our design.

5. System Evaluation

We evaluate the Fast-Fi system by first assessing its performance under common usage scenarios as well as its scalability to campuses larger than that at MIT. We next justify the throughput prediction model and discuss connection time and latency implications. Finally, we address some of the security vulnerabilities of the system and propose solutions to these issues that make sense in certain implementations.

5.1 Use Cases

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

5.1.1 Single Client, Underutilized Network

When a solitary client is in an underutilized part of the network, it is likely that the client will initially connect to an AP that is underutilized. In this case, the connected AP suggests itself as a satisfactory (as defined in Section 3.3) connection point, and will immediately begin serving the client's traffic. If the client connects to an AP that is unable to handle all of the client's traffic, it looks for another in-range AP (provided by the client) that is able to handle the client's traffic. Since the client is in an underutilized part of the network, there should be little competing throughput. Therefore, time required for the client to join the network is very small.

5.1.2 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 accommodate all the clients' traffic, then the mechanism of searching when dissatisfied results in load-balancing that usually leads to a stable distribution of users (discussed further in Section 5.2.1).

On the other hand, different clients may have access to different APs. The fraction APs that are in-range of more clients are liable to become throughput-limited, putting their clients' controllers in a *dissatisfied* state. At this point, clients in range of less congested 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 near-full utilization will be achieved as connecting clients attempt to maximize their throughput. Meanwhile, clients will each be recommended the location of another satisfactory AP within 500 ft.

5.1.3 Many Moving Clients

In regions of high churn, such as a hallway, devices on 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 that 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. As future work, we believe that analyzing the geographic movement of client could be beneficial for predicting connection patterns. That is, if a student is walking down a hallway, we could predict the next AP the student will pass, and prioritize it when choosing among the next possible APs. If we can successfully predict client movement patterns, there would be fewer transient connections, which would increase both user happiness and network performance.

5.2 Scaling Beyond MIT

Fast-Fi is robust to changes in scale with respect to both users and APs. Where possible, the system pushes work out to the APs to minimize the load and dependency on the central server, lending itself very well to scaling to large networks.

5.2.1 Scaling with Respect to Users

Upon an influx of users connecting to the network, Fast-Fi's mechanism of searching for satisfactory APs when users are unhappy results in eventual stability if a stable configuration exists and the APs are not all near maximum throughput capacity. Although an AP may have more clients connected to it than it can handle initially, any of its "unhappy" clients will cause it to continually search for other APs that can better serve the clients' needs. (Recall that unhappy clients are those who either have less throughput than they would like or those who have pressed the unhappy button.) Thus, as long as there are nearby APs that can provide sufficient throughput for unhappy clients, the unhappy clients will be reconnected to these APs that can provide better connections.

As the number of users increases to the point of AP throughput saturation (in which APs can no longer satisfy the clients' throughput demands), Fast-Fi trades off optimal utilization for user happiness. While an alternative solution may reshuffle clients by solving a knapsack-like optimization problem, we decided that the performance costs of enforcing this policy outweighed the benefits. Not only would this policy add a large degree of complexity to the system, users already connected to a network would have to wait to disconnect and reconnect (thereby decreasing their happiness). Furthermore, the knapsack problem may not even have a solution if the network is very saturated. Thus, while Fast-Fi may not achieve optimal utilization in the edge case of near-AP-saturation, we trade off this edge-case performance for simplicity and user happiness.

5.2.2 Scaling with Respect to APs

Due to the Fast-Fi's mostly distributed nature, new APs can easily be added (through the registration process described in Section 2.4). After joining the network, this new AP receives a list of nearby APs (within $D = 625$ ft of itself) from the central IS&T server. Then, within six hours, any nearby APs will have requested updated information from the IS&T server and will have been informed of this new AP.

The size of these updates is $2 \cdot (n_D - 1) = \Theta(n_D)$, where n_D is the number of APs packed in a circle of radius D around the new AP. This is because the IS&T server 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 component 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 from the APs (i.e. bytes transferred and number of clients connected) is compressed and the daily files are relatively small (under 6.5MB each, as per Section 3.5) but needed to facilitate network analysis. The increase in storage requirements on the IS&T server scales linearly with the number of APs added to the system, and there is no way to make this more efficient since we need to collect usage data for each of these new APs.

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.

5.3 Throughput Prediction

We believe that our model for predicting throughput for a client joining an AP will work well in practice. By projecting the state of the network a few moments into the future, the Fast-Fi system can readily predict and prevent a client from joining a network that will soon suffer from poor performance.

As discussed in Section 3.2.2, the system predicts a client's expected throughput after joining the network by predicting the number of clients and the total throughput at the AP. Instead of using the instantaneous values, the AP fits a regression (linear, low-order polynomial, logistic, or exponential) to previous data points, allowing the system to give a good estimate of network conditions during the time immediately *after* a user connects to the network.

We note that if AP hardware is upgraded in the future, more complex signal analysis techniques may be employed for throughput prediction, and may take into account daily or weekly trends (e.g. lecture schedules) since more data could be stored and analyzed to make more accurate predictions. These techniques might include Fourier analysis and/or gradient descent to tune weighted combinations of various curves, methods we believe are too computationally intensive to run on the current AP hardware.

An example that demonstrates the power of this prediction model can be given by considering the use case in which many students enter a lecture hall (such as 26-100 at MIT) in the minutes leading up to the lecture. A few students will enter very early, most students will enter within a few minutes before the lecture, and then some stragglers will enter just on time or slightly after. The logistic curve given in Figure 4 illustrates this flow of students entering the classroom.

While a logistic regression (or perhaps a higher-order polynomial regression) would work well in the classroom example given above, there are cases where certain models would not work well, and in fact, may cause degraded performance due

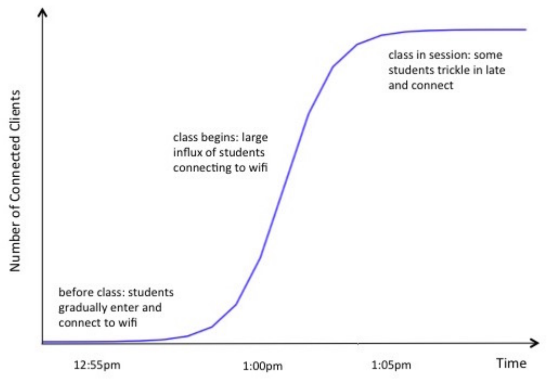

Figure 4. A logistic regression modeling flow of students into a large lecture hall before a class. About ten minutes before lecture, students will slowly start entering at an increasing rate, with most students entering a few minutes before. After the lecture begins, some late students will enter at a decreasing rate.

to poor predictions. For example, in the use case of many users walking down a hallway (such that devices have high turnover among APs), an exponential regression would likely fit very poorly, and with a high update frequency would probably lead to issues of over-fitting. By fitting to relatively simple models, we alleviate issues of over-fitting the network data.

In the case where no regression model fits the data well (the client connection and/or throughput data are seemingly random), a constant or linear curve would likely end up fitting around this noise, and then throughput prediction would just be estimated based roughly on the average throughput over the previous time points, which would be sufficient for a client controller seeking conditions about the network.

Compare this against a system that does not use predictive throughput. In the time when the rate of users connecting to APs in the classroom is maximized (the middle section of the curve in Figure 4), these devices can potentially all join only a small number of APs in the classroom. The AP will likely suffer from poor performance because of the large number of users and their throughput demands, and many users will be unhappy with the network conditions. In Fast-Fi, the throughput prediction allows devices to be distributed among APs that have better expected network conditions. When many users start joining a particular AP (i.e. it predicts that a few time steps into the future it will have even more users and consequently reduced expected throughput), that AP will advertise a very poor expected throughput to other devices looking to join. Instead, if there is an AP that has not been inundated with new devices connecting (this may happen if it is slightly further away in the classroom or has lower signal strength), it can provide better throughput, so users would be redirected to this AP. Once users start connecting to this AP, the prediction model realizes that expected throughput will go down, and then this cycle continues until the network load is distributed among the APs in the classroom.

We believe that an update frequency of 1 Hz for APs to refit their regressions is sufficient and provides a compromise between having up-to-date usage information about nearby APs and not wasting network resources and computing power to request data from other APs and refit the regression. To address the issue of network performance in particular, each AP caches the values from nearby APs with an expiry time of 1 second so that a network does not send any unnecessary requests to nearby APs in the event that many clients are looking to connect. Similarly, to address any issues associated with the computation overhead required to refit the regression model every second, depending on which model is being used, there are online algorithms that make refitting the model quicker. This technique is fairly straightforward for linear models but becomes tricker for higher order and exponential models.

5.4 Connection Time

Next, we will evaluate the process of establishing an internet connection with a Fast-Fi AP. First, the client controller searches for in-range APs. To avoid anomalous delays from thoroughly scanning every channel, after seeing one AP the client continues scanning only for a short time $t_{scan} = 0.385$ seconds, then initiates its connection with whatever APs it sees in this time. This is valid because if any of these visible APs can serve the client's traffic, then the protocol has succeeded perfectly, and conversely if none of them can serve it well, we connect anyway and rely on the client's periodic checking for a better AP to allow it to find a satisfactory in-range AP eventually. t_{scan} must be chosen carefully, trading off connection time for throughput; however, with 11 channels to scan, 5 ms seconds to switch between channels, and 30 ms max time to listen for a heartbeat, scanning every channel should take no more than 0.385 seconds.

After the client initiates communication with the first AP, the AP must converse with the visible APs in order to ascertain their throughputs. In order to evaluate the time cost of this operation, we must estimate the roundtrip latency of this communication t_{rtt}. In an empirical test conducted on the MIT wired network, we observed a latency of 0.5 ms over a 8-hop link. Geographically close APs should not be more than 8 hops apart in an average usage scenario (e.g. in the interior of a building, where most visible access points are all connected to the building's router), so 0.5 ms is a good estimate of the latency of our system. Because these requests can be dispatched all at once and waited for in parallel, 0.5 ms is our bound for this step.

Finally, the client establishes a connection with the recommended AP. This is identical to connecting to standard AP in a non-Fast-Fi-enabled system, equal to a small number of seconds t_{std}.

Thus, we get a final connection time

$$t_{FF} = t_{scan} + t_{rtt} + t_{std} = t_{std} + 0.385\text{sec}.$$

Thus, the connection time for Fast-Fi should only be about half of a second second longer than a standard system. Because normal connection times are on the order of several seconds, and because we expect returns in throughput from scanning longer for APs, this amount is very reasonable, and likely not noticeable for the user.

5.5 Communication Overhead

We now discuss the communication overhead in the Fast-Fi system. First, we analyze the overhead for communication between APs and then the overhead between APs and the central IS&T server.

5.5.1 AP-AP Communication Overhead

Fast-Fi APs use the wired network to exchange throughput information. We will consider the worst-case performance of this system. Consider a campus building that contains 100 APs. Because these APs know each others' IP addresses and are connected via the building subnet, none of their traffic need leave this building, so we only need to consider whether this building LAN itself can accommodate their traffic. Under conditions of maximal network use, the number of AP-AP requests they send will be limited by the 1 Hz caching of these requests. If we make the further assumption that 50 APs asked within 625 feet of an AP before recommendation, we yield an extreme bound on their overhead:

$$50 \text{ req/sec/AP} \times 100 \text{ APs} \times 100 \text{ B/req} = 0.5 \text{ MB/sec}$$

The 0.5 MB/sec is well within the 1 Gbps capacity of the network; thus, there is a great deal of scalability in AP density that could be accommodated by this system.

5.5.2 AP-IS&T Server Logging Communication Overhead

As described in Section 3.5, each AP collects usage information for 24 hours and then sends this compressed binary dump to the IS&T server. Therefore, it would take at most 24 hours for any real-time usage information to be imported into the SQL database on the IS&T server.

In terms of the time required to transmit the dump file, we know that the compressed format allows the binary file to be small (at most 6.5MB, as computed in Section 3.5). Therefore, since the latency between the APs and the IS&T server on the wired network is no worse than 10 ms (it's usually much closer to 1 ms), and the wired network supports throughput up to 1Gbps (125MB/sec), it would take much less than a second to transmit the file. Accounting for any additional time required to establish the SFTP session, even if the file transfer took one second, this time is still very small and very reasonable for our system.

5.6 Security

We consider the Fast-Fi design in terms of security vulnerabilities. Since all usage history data stored for logging at the IS&T server is compiled as aggregate traffic information on each AP once per second, it is not possible for an adversary to determine which users were connected to which APs solely from this data. No MAC address information about any connected devices is stored or logged on the server.

However, there are other potential security vulnerabilities in the system. For example, since communication between APs and between an AP and the IS&T server is neither authenticated nor encrypted, it is possible for an adversary to execute a man-in-the-middle attack. In such an attack, the adversary could pretend to be an AP to send false or malicious data to other APs and/or the IS&T server, which could compromise network performance and integrity. By sniffing packets on the network, the adversary could easily determine the communication protocol in the system, and then use this knowledge to execute the attack.

An implementor of this system could easily fix these vulnerabilities by authenticating and encrypting all communication between the APs as well as APs and the IS&T server. There are standard mechanisms for solving this problem, such as SSL.[1]

Encrypting and authenticating all system communication would both prevent an adversary from learning the protocol, as well as prevent the adversary from spoofing an AP or the IS&T server on the network, since the adversary would be unable to produce a valid authentication token as that device. However, there is a large computational and performance overhead in doing so. The APs and the IS&T server need to first generate and exchange an RSA key pair (one-time overhead). Then for each communication session, they need to first transmit a few messages to generate an encryption key, which takes time and would increase communication latency between the endpoints. Future packets transmitted may also have larger size because of padding added by the encryption algorithms.

For this reason, we elect not to authenticate and encrypt communication in the default system, in order to facilitate maximum performance. However, as shown, the system design is scalable and modular in that it would be simple to add the authentication and encryption layers on top of the current communication protocol without significant change to any individual components of the system.

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

[1] SSL uses the public key of an RSA key pair generated at all end points to generate and exchange an ephemeral key (thereby authenticating the end points, since only the intended end point has the secret key necessary for decryption). This ephemeral key is then used for encrypting message data for the remainder of the session using a symmetric encryption algorithm such as AES (thereby ensuring that an adversary cannot decrypt or forge any communication in the system).

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, fault-tolerant, and modular solution as required by the network's users.

Acknowledgments

We would like to thank our instructors Matei Zaharia, Juergen Schoenstein, and Cathie Yun for their help and advice. The project would not have been as successful without their assistance.