Analysis of wireless information locality and association patterns in a campus

Francisco Chinchilla, Maria Papadopouli
Department of Computer Science, The University of North Carolina at Chapel Hill

***Abstract*— Our goal is to explore characteristics of the wireless environment that provide opportunities for caching, prefetching, coverage planning, and resource reservation. We conducted a one-month measurement study of locality phenomena among wireless web users and their association patterns on a major university campus using the IEEE 802.11 wireless infrastructure.**

We evaluate the performance of different caching paradigms, such as single user cache, cache attached to an access point (AP), and peer-to-peer caching. In several settings such caching mechanisms could be beneficial. Unlike other measurement studies in wired networks in which 25% to 40% of documents draw 70% of web access, our traces indicate that 13% of unique URLs draws this number of web accesses. In addition, the overall potential ideal hit ratios of the user cache, cache attached to an access point, and peer-to-peer caching paradigms (where peers are coresident within an AP) are 51%, 55%, and 25%, respectively.

We distinguish wireless clients based on their inter-building mobility, their visits to APs, their continuous walks in the wireless infrastructure, and their wireless information access during these periods. We model the associations as a Markov chain using as state information the most recent AP visits. We can predict with high probability (86%) the next AP with which a wireless client will associate. Also, there are APs with a high percentage of user revisits. Such measurements can benefit protocols and algorithms that aim to improve the performance of the wireless infrastructures by load balancing, admission control, and resource reservation across APs.

I. Introduction

Recently IEEE 802.11 networks became widely available in universities and corporations to provide wireless Internet access. There are very few studies investigating the information access via the wireless infrastructure, the performance of the wireless infrastructure, and the user access and mobility pattern. Research on these issues can impact several areas, such as simulation studies on wireless networks, deployment and administration of wireless infrastructures, protocol design for intelligent and robust wireless infrastructures, and user access and traffic characterization. Currently, most of the simulation studies on wireless networks and protocols consider simplistic communication and association patterns for the wireless users. There is a need for more realistic models of the user communication and association patterns to drive simulations on wireless networks. The wireless clients associated with an access point (AP) share the buffer, cache, and bandwidth at that AP. Insights from traffic characteristics at each AP and the wireless users transitions from one AP to another can assist in deploying the content distribution network, capacity planning and wireless network deployment, and development of admission control and reservation allocation mechanisms. This is important not only for preventing and resolving user congestion, but also for the provision of quality of service guarantees for the support of real-time streaming, voice over IP, location-dependent, augmented-reality, and other applications with strict response delay requirements.

Mobile users experience frequent loss of connectivity and high end-to-end delays when they access the wireless Internet. In an earlier work [7], we investigated a peer-to-peer approach that introduced a new paradigm of information sharing and cooperating caching among mobile devices not necessarily connected to the Internet. This paradigm exploits the *spatial locality of queries and information* of mobile users. An environment is characterized by spatial locality of queries and information when users in close geographic proximity are likely to query for the same data. In such environment, we considered the role of stationary caches in wireless LANs and also thin cooperative caches of mobile devices (i.e., peers). When a wireless device is unable to access the data via the Internet, it can acquire the data from these caches. We showed via simulation that in settings with high spatial locality of information and frequent disconnections from the Internet, these peer-to-peer systems can enhance the information access by reducing the average delay to receive the data.

We want to explore characteristics of the wireless infrastructure that provide opportunities for caching, prefetching, coverage planning of wireless infrastructure, and resource reservation. The web provides a ready testbed to study the prevalence of information locality and access. Since wireless association and user mobility are two new characteristics of the environment, we would like also to investigate their impact on the information access. The main goal is to provide more robust and intelligent wireless infrastructures by understanding the wireless web access and user association patterns.

The UNC wireless infrastructure provides coverage for nearly every building in the 729-acre campus and includes a diverse academic environment of university departments, programs, administrative, activities, and residential buildings. In these buildings, there are 26,000 students, 3,000 faculty members, and 9,000 staff/administrative personnel. Of the 26,000 students, 61% are undergraduates, and more than 75% of these own a wireless laptop.

The three key issues that drive this study are:

Temporal and spatial locality of information. The temporal locality identifies the frequency and temporal aspects of repeated requests for some information. The spatial locality focuses on the AP and building in which a repeated request occurs and indicates if the repeated request originated from a nearby client, a client within the same AP, or a client in the same building.

Caching paradigms: *user cache, cache attached to an AP or a building, and peer-to-peer caching*. The user cache is considered to be the web browser cache. The cache attached to an AP or a building will serve the wireless clients associated to that AP or to APs of that building, respectively. In the peer-to-peer caching clients in wireless range act as cooperative thin caches for each other.

User association and mobility patterns. We distinguish wireless clients based on their inter-building mobility, their visits to APs, their continuous walks in the wireless infrastructure, and their wireless information access during these periods.

We differ from previous studies on the wireless infrastructure ([5], [1], and [8]) by focusing on the wireless information locality. To our knowledge, this is the first empirical study on the information locality via the wireless infrastructure. Unlike previous studies of wireless networks, we explore the impact of the wireless information locality and feasibility of three main caching paradigms. This research also extends the studies by Kotz and Essien [5], Balachandran *et al.* [1], and Tang and Baker [8] by focusing more closely on the association and mobility patterns of individual clients rather than on the entire population of mobile clients and in a finer time granularity. Unlike the Dartmouth wireless infrastructure [5], the UNC wireless clients maintain one single IP address throughout their roaming in the wireless infrastructure and keep the same ID (based on its MAC address) throughout the entire trace. This allows us to correlate the wireless users with their web access and association patterns and carry out user-behavior analysis more accurately.

A. Summary of main contributions

Several studies on web caching over the wired network (such as [3]) report that the distribution of web requests from a fixed group of users follows a Zipf-like distribution. Overall, the URL and web-server popularity distribution in our traces are modeled well as a Zipf-like function (with α_{zipf} equal to 0.85 and 1, respectively) [6]. of web accesses. In a previous study, 25% to 40% of documents draw 70% of web access [3]. However, our traces indicate that only 13% of unique URLs draws this percentage of web accesses.

This can be defined as our *ideal hit ratio* that can be achieved assuming an infinite cache and that shared documents are cacheable. Another empirical study in the wired infrastructure in a university campus reports a 45%-59% ideal hit ratio for a similar user population size [9].

The overall ideal hit ratios of the user cache, cache attached to an AP, and peer-to-peer caching paradigms (where peers are coresident within an AP) are 51%, 55%, and 25%, respectively. By *overall ideal hit ratio*, we mean the percentage of the total repeated requests that have occurred throughout the HTTP tracing period. As in several wired traces studies, the single-client locality is the primary factor in our wireless traces. We show that there is opportunity for improving the wireless access by more actively caching data in the user cache. The high temporal locality of data can enable prefetching systems and replacement algorithms that maximize the data availability over short time periods and minimize the extended storage of data when it has not been recently requested. Similarly, the probability that a client will revisit an AP can provide some guidelines for the expiration of objects related to that client in the AP cache. More specifically, each AP observes a number of revisits from wireless clients that had associated with it during the last hour. We computed the fraction of visits that are revisits for each AP and client and observed that some APs and clients have a high revisit probability. There are APs with a 95% percentage of revisits out of all visits to that AP. However, the revisit probability varies drastically among both APs and clients.

We model the associations as a Markov chain using as state information the most recent AP visits. We are able to predict with high probability (86%) the next AP with which a wireless client will associate. This type of predictions can benefit several protocols and algorithms that aim to improve the performance of the wireless infrastructures by load balancing, admission control, and resource reservation at APs.

Section II of this paper describes previous related research; Section III explains our techniques for acquiring the data for this study; Section IV describes our analysis on the locality of the web URLs via the wireless infrastructure and presents different caching paradigms and their performance. Section V provides insight about the associations and movement of users on the campus and discusses their access pattern. In Section VI, we summarize our main results and discuss future work.

II. Related work

Collaborative caching among conventional clients in the wired network has been analyzed by Duska *et al.* [4]. They find the benefit of such caching to be limited by the diversity of clients' requests and the non-cacheability of many objects.

We examine the collaborative caching among wireless clients by focusing on clients within the range of an AP as well as other caching paradigms. We compare our results on caching with other measurement studies on the wired network [9], [3] in Sections IV and I-A.

There have been other studies of client mobility and access patterns [5], [1], [8]. Bhattacharya and Sajal [2] performed a study of client mobility patterns using a PCS network, and propose a prediction mechanism based on Markov states. Balachandran *et al.* [1] performed similar measurements in a three-day conference setting, also focusing on the offered network load, and global AP utilization. They characterized wireless users and their workload and addressed the network capacity planning problem. Both [1] and [8] analyze a setting and data traces that are different from ours. The study closest to ours is the one done by Kotz and Essien [5] that characterized Dartmouth's wireless network, examining global traffic and AP utilization. We contrast our results with the Dartmouth study in detail in Section V. Moreover, we build on the work in these papers by directing our attention to web traffic. Instead of looking at global patterns of mobility, we focus efforts on modeling an individual user. Unlike any of these studies, we also examine the locality of information and caching.

III. Data Acquisition

Two sets of data were used in this study: traces of wireless clients' web requests and logs of 802.11 MAC events generated by wireless APs in the campus. These sets were correlated using their timestamp information and thereby allowing us to determine the AP from which each web request was made.

A. Definitions

The campus is populated by people who have devices that communicate with the campus wireless network; each such device is called a client. Each client has a unique MAC address, and is assigned a (positive) unique client ID number based on its MAC address (see Section III-D for details on the techniques used to ensure privacy).

The campus has many APs, each of which is a non-moving bridge between the conventional campus network and the wireless network. Since each AP has a unique IP address, we use an AP's IP address to determine its (positive) unique AP ID number. Each AP has a coverage area determined by radio propagation properties around the AP. A client communicates via the network by associating itself with an AP; we say synonymously that such a client visits the AP. For a more detailed description on how a client associates with an AP, see Section III-E.

B. Campus wireless network

The University of North Carolina at Chapel Hill began deployment of an IEEE 802.11 network in 1999. Wireless access is available in many residence halls, academic buildings, the medical school, and in some off-campus administrative buildings. The campus uses primarily Cisco Aironet 350 802.11 APs, although some areas on campus are serviced by older APs from other manufacturers. We observed 7,681 distinct wireless clients during the tracing period February 10 through April 27, 2003, and observed 2,879 wireless clients making one or more HTTP requests during the tracing period February 26 through March 24, 2003.

C. HTTP traces

The bulk of the campus wireless network has a single aggregation point that connects to a gateway router. This router provides connectivity between the wireless network and the wired links, including all of the campus computing infrastructure and the Internet. We connected to a monitor port on the gateway router, letting us monitor all of the traffic that passed between the wireless network and conventional wired networks. This tap link was connected to a FreeBSD monitoring system. We used the tracing tool tcpdump to collect all TCP packets that have payloads that begin with the ASCII string " GET" followed by a space. The full frame was collected as a potential HTTP request. We did not restrict our collection to the standard HTTP port, allowing us to record HTTP requests sent to servers on non-standard ports, which include many common peer-to-peer file-sharing applications. The packet trace was then processed to extract the HTTP GET requests contained therein. From each packet, we kept these items:

the *time of the packet's receipt* (with one-second resolution), the *hostname specified in the request's Host header*, the *Request-URI*, and the *hardware MAC address* of the wireless 802.11 client.

If all of these items were not available in a packet, then we did not include the recorded packet in our recorded requests. Using these criteria, 8,358,048 requests for 2,437,736 unique URLs were traced and included in the analysis. By recording the traffic before it had passed through an IP router, we were able to capture the original MAC header as generated by the 802.11 clients for transmission to the gateway router. To avoid violating our users' expectations of privacy, we do not store the hostname, path, and client MAC address directly.

D. Privacy assurances

To avoid disclosure of the identity of individual users and of the sites that a user is visiting, we store and use SHA1 hashes of the client's MAC address, the request hostname, and the requested path. The MAC address uniquely identifies an 802.11 network device; we assume it to be coupled to a specific computer. Two requests are considered to be from the same client if they were generated by clients that have the same hashed MAC address, and two requests are considered to be for the same URL if they have the same hashed hostname and request path.

E. Access point logs

The majority (232) of the APs on campus were configured to send syslog events to a server in our department between 12:00:00 am on February 10, 2003 and 11:59:59 pm April 27, 2003. During this trace period we recorded 8,158,341 syslog events for 7,694 clients, and 222 APs distributed among 79 buildings. The following definitions and rules closely follow those in Section 3 of the Dartmouth study [5].

1) Syslog events: There are seven types of events that trigger an AP to transmit a syslog message. These messages and their corresponding events are interpreted as follows:

Authenticated: A card must authenticate itself before using the network. Since a card still has to associate with an AP before sending and receiving data, we ignore any authenticated messages.

Associated: After it authenticates itself, a card associates with an AP. Any data transmitted to and from the network is transmitted by the AP.

Reassociated: A card may reassociate itself with a different AP (usually due to higher signal strength). After a reassociation with an AP, any data transmitted to and from the network is transmitted by the AP.

Roamed: After reassociation occurs, the old AP sends a roamed message as well as the AP with which the card has just reassociated. Since we still receive the reassociated message, we can ignore this message as well.

Reset: When a card's connection is reset, a reset message is sent. In our trace, cards with a reset message are only involved in reset messages. We believe this to be an artifact of us not having logs from all of the APs, and therefore ignore any reset messages.

Dissasociated: When a card wishes to disconnect from the AP it disassociates itself. We ignore any disassociated messages for a card if the previous message for that card was a disassociated or a deauthenticated message.

Deauthenticated: When a card is no longer part of the network a deauthenticated message is sent. It is not unusual to see repeated deauthenticated messages for the same card, with no other type of events for that card in between. We ignore any deauthenticated messages for a card if the previous message for that card was a disassociated or a deauthenticated message. A disconnection message describes either a disassociated or deauthenticated message.

2) Visits and sessions: Using the events as described above, we need to define what visits and sessions are, as well as some of their properties such as duration time. We assume that each event occurs at the time of the timestamp in the corresponding syslog entry. The exception is that if a client is deauthenticated due to an inactivity period of 30 minutes (or more), we consider the disconnection to have occurred 30 minutes before the timestamp that appears in the corresponding deauthenticated syslog entry.

Events: Only the associated, reassociated, deauthenticated, and disassociated events as mentioned above are considered.

State: A state represents the AP with which client is currently associated with. When a client is connected to the network, its state is the numeric ID of the AP it is currently associated with (via an association or a reassociation). When the client is disconnected from the network, then its state is defined to be "0". Since we do not know where the clients are before the trace begins, each client is considered to be in state 0 at the beginning of the trace.

State history: The state history of a client is the ordered sequence of *states* that the client has visited.

Reconnection threshold: Sometimes a client will disassociate or deauthenticate for a single second and then associate or reassociate. Whenever a client is disconnected for one second or less, we do not consider the client to have disconnected from or left the network, but instead consider this to be part of the reconnection procedure. We believe this to more accurately represent the user's intentions.

These rules leave us with 2,389,066 useful syslog events for 6,186 clients and allow us to define the following terms:

Visit: A client begins a visit to AP when a (re)association message is received from that AP for that client and ends when *any* message from *any* AP is received for that same client. The difference in the timestamp of these two messages defines the duration of the visit.

Session: A session is a sequence of visits to APs. A session begins when a currently disconnected client receives a (re)association message and ends when the next disconnection message is received. The difference in the timestamps between the disconnection message and the first (re)association message defines the duration of the session. A session can be mobile, roaming, or a visit.

Inter-AP transition: If a client is currently associated to an AP, an inter-AP transition is defined as a (re)association to a *different* AP. The two APs may or may not be in the same building.

Inter-building transition: If a client is currently associated to an AP at a certain building, an inter-building transition is defined as a (re)association to an AP located in a different building.

Roaming session: A roaming session is a sequence of consecutive visits (with no disconnections) that includes two or more distinct APs. A roaming session begins when a currently disconnected client receives a (re)association message and ends when the next disconnection message is received. The difference in the timestamps between the disconnection message and the first connection message defines the duration of the roaming session.

Mobile session: A mobile session is a special type of roaming session that comprises of a sequence of consecutive visits (with no disconnections) to two or more different buildings. A mobile session begins when a currently disconnected client receives a (re)association message and ends when the next disconnection message is received. The difference in the timestamps between the disconnection message and the first (re)association message define the duration of the mobile session.

Roaming client: A client with a roaming session is called a roaming client.

Mobile client: A client with a mobile session is called a mobile client.

Drop-in client: A drop-in client is a card that visits two or more buildings in the period of time in question. Drop-in clients may have disconnections in between the visits to these buildings.

Event type	**Events**	**Clients**	**APs**	**Buildings**
Total syslog	8,158,341	7,694	222	79
Useful syslog	2,389,066	6,186	222	79

TABLE I

SUMMARY OF SYSLOG STATISTICS

F. HTTP requests model

We use a post-processing phase that conceptually examines every request in the HTTP trace and identifies the AP via which it was made using the syslog trace. We correlate the HTTP requests in the HTTP trace with the messages of syslog trace as follows: Let us assume that in the HTTP trace an HTTP request appears as (u, t, c), where u is the request URL, t is the time of the packet's receipt at the sniffer, and c the wireless client id. For each HTTP request, we parse the syslog entries to find the last association or reassociation message for the client with a timestamp earlier than the HTTP request time. If we find such a message, we assume that this is the AP p from which the client requested this data and generate a request of the form (u, t, c, p). For the next sections, we assume that we have a stream of these requests in increasing order using their timestamp. The URL identifies the request object.

IV. LOCALITY OF WEB OBJECTS

Web requests may exhibit different locality characteristics. We will classify them into *same-client*, *same-AP*, *AP-coresident client*, *same-building*, and *campus-wide* repeated requests. The above classification of the locality is hybrid, in that it exhibits both temporal and spatial characteristics. In the following sections, we discuss them and present the locality that our client requests have.

A. Same-client repeated requests

A same-client repeated request occurs when a single client requests an object that it has requested in the past. The cause could be any of these:

Subsequent request. A client intentionally requests an object that it has requested in the past, but could not be satisfied by the browser cache. Such a request would represent genuine ongoing interest by some client.

Client reloads. A client reloads a page. This may occur when the page has not been transmitted properly.

Automatic reloads. Many popular pages (such as headline-news and weather sites) cause the browser to reload the page periodically. While the page is displayed, the browser will periodically re-request it. Some of these requests could also be considered indicative of continued interest by the client.

Packet retransmissions. If the first packet containing the

Symbol	**Description**
Request representation r	$r(u_r, t_r, c_r, p_r)$
t_r	time when request was received from sniffer
u_r	URL or the request r
c_r	client sent request r
p_r	AP via which request r was sent
C	total number of clients in the http trace
R	the stream of requests in the entire HTTP trace
B	set of buildings
C^u	set of distinct clients that have requested URL u
B_i^u	set of buildings from which client i requested u
$S_i(t)$	state of client i at time t (AP id or "0")

TABLE II

NOTATION FOR LOCALITY OF INFORMATION.

request was not known by the client to have reached its destination, TCP specifies that the client retransmit the packet. We record both requests as distinct requests. However, we expect that such retransmissions are rare [6].

This study is subject to the effects of browser caching; if the requested object is in the browser's cache, then no HTTP request will be generated. Some, but not all, browsers follow HTTP's specification for determining the freshness of a cached object. Also, we speculate that a percentage of the repeated requests are conditional HTTP GET requests. This measure does not account for the location of the client and therefore reveals temporal but not spatial locality. We compute the temporal locality of these requests as follows: For each request in the trace (such as $r(u, t, c, p)$ for a URL u made at time t by client c via AP p), we check backwards in time for previous references to the same URL u made by the same client c. If such request $r'(u, t', c, p')$ is found, we record the time that has elapsed since this request occurred, $t - t'$. The set of same-client repeated requests for an time interval w is $SC(w)$, where

$$SC(w) = \{\forall\, r(u,t,i,p) \in R \mid \exists r'(u,t',i,p') \in R \wedge t - t' \leq w\,\}.$$

B. Same-AP repeated requests

When an object is requested multiple times within the same AP's range, those are called same-AP repeated requests. This measure does not account for the client that makes the request; i.e., the repetition can occur due to a single client or several clients requesting the same object within a single AP's range.

We compute the same-AP repeated requests as follows: For each request in the trace, $r(u, c, t, p)$, we check backwards in time for previous references to the same object u accessed from the same AP p. If such request $r'(u, t', c', p)$ is found, we record the time that has elapsed since that request occurred, $t - t'$. The set of same-AP repeated requests within a time interval w, $SA(w)$, is given by

$$SA(w) = \{\forall\, r(u,t,i,p) \in R \mid \exists\, r'(u,t',i',p) \in R \wedge t - t' \leq w\}.$$

C. AP-coresident-client repeated requests

At the heart of measuring spatial locality effects among mobile web users is this: How often are users who are interested in the same things near one another? We answer this question by examining object and client-AP as well as object and client-building correlations. These spatial locality properties of wireless web access can impact caching.

An AP-coresident client repeated request is said to occur when a client in an AP's area requests an object that has been requested at some time in the past by another client who is in the same AP's area at the time that the new request is made. Note that this other client that requested the object in the past, may have requested the object while at a different location. This indicates that two different clients have requested the same object and were near one another at time of the second request.

For each request in the trace $r(u, c, t, p)$, we check backwards in time for previous references to the same object u made by a client c' that is currently at the same AP p. If such request $r'(u, t', c', p')$ is found, we record the time that has elapsed since this request occurred, $t - t'$. We compute the set of such requests $AC(w)$ that occur within the time interval w as

$$AC(w) = \{\forall\, r(u,t,i,p) \in R \mid \exists\, r'(u,t',i',p') \in R \wedge t - t' \leq w \wedge S_{i'}(t) = p\ \}.$$

Figure 1 displays the fraction of same-client (f_{SC}), same-AP (f_{SA}), and AP-coresident (f_{AC}) client repeated requests

$$\frac{|SC(w)|}{\sum_{i \in C} R_i}, \frac{|SA(w)|}{\sum_{i \in C} R_i}, \text{ and } \frac{|AC(w)|}{\sum_{i \in C} R_i},$$

respectively, for an interval w equal to one hour. More specifically, the fraction of repeated requests at each minute t in Figure 1 are the **additional** repeated requests

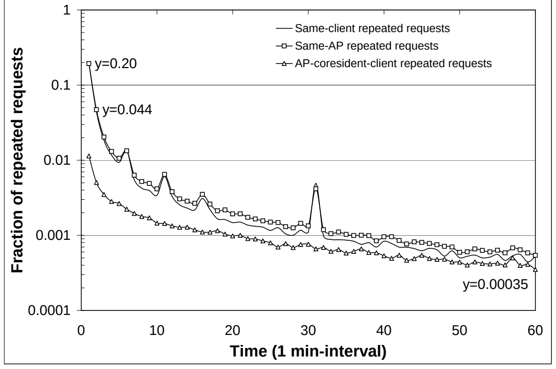

Fig. 1. Fraction of repeated requests within a one-hour interval. The number of requests considered is at least 7.6 million. Over 2,800 clients are represented.

that occur in that minute of the first hour. For example, within the first minute the fraction of repeated requests are at least 0.19 for same-client (i.e., $f_{SC}(1\text{ min})$) and same-AP (i.e., $f_{SA}(1\text{ min})$) and 0.01 for AP-coresident client (i.e., $f_{AC}(1\text{ min})$). In the second minute, an *additional* 0.04 fraction of requests are same-client (i.e., $f_{SC}(2\text{ min})$-$f_{SC}(1\text{ min})$) and same AP (i.e., $f_{SA}(2\text{ min})$-$f_{SA}(1\text{ min})$) repeated requests, and the fraction of *additional* repeated requests is 0.005 for AP-coresident client repeated requests (i.e., $f_{AC}(2\text{ min})$-$f_{AC}(1\text{ min})$). The plot shows that there is periodicity in same-client and same-AP repeated requests after 5, 10, 15, and 30 minutes. In AP-coresident client repeated requests, however, there is no such correlation in the web access patterns of clients. We find that the fraction of repeated requests for same-client and same-AP are similar and higher than that of AP-coresident repeated requests. As many as 37% of all requests would be unnecessary if every object on the web had a cache lifetime of at least an hour. This indicates the impact of the client's web browser cache, assuming that all browsers observe the HTTP standard for caching. The repeated requests follow a power law with exponent coefficients of -1.31, -1.27, and -0.76 for same-client, same-AP, and AP-coresident client, respectively. The coefficient of determination (R^2) is at least 0.94 for all of them. These coefficients indicate that the temporal locality is more apparent in the same-client but not in the AP-coresident client caches.

We also computed the same-user repeated requests considering only the clients with high and low number of inter-building transitions. Of the top five percent (309 clients), 240 request at least one web object and reach 52% same-client repeated requests with a mean and median of 40% and 41%, respectively. We also selected the 309 wireless clients that stayed within one building during the entire trace period and had the highest number of associations. Only 45 of these clients requested at least one web object, and these reach 55% repeated requests with a 42% mean and 40% median. Figure 2 shows the fraction of repeated requests for an interval w equal to the entire trace. For example, within a day the percentage of repeated requests is 44% for same-client, 48% for same-AP repeated requests, and only 15% for AP-coresidential client repeated requests. In the second day, an **additional** fraction 0.02 of requests are same-client and same AP repeated requests, and the fraction of repeated requests is 0.03 for AP-coresident client repeated requests.

Our results are conservative because they include compulsory (cold start) misses. We minimize this effect by taking measurement traces over 26 days. On the other hand, we assume infinite cache and that shared documents are cacheable. Therefore, the following hit ratios are ideal hit ratios. A cache at each AP would achieve an ideal hit ratio of 55% for the whole trace, whereas a cache that serves the entire campus would achieve an ideal hit ratio of 71%. There are APs with higher ideal hit ratios; for example, an AP in an auditorium had an ideal hit ratio of 73% that corresponds to the 40,064 requests made by six distinct users. We observe that 7% of all requests are for objects that have been requested by a nearby client within the last hour. Furthermore, this proportion varies widely; at some locations on the campus, 15% of all requests were for such objects. Also, lower number of HTTP requests and fraction of repeated requests are made on weekends than on weekdays [6], and several repeated requests exhibit 24-hour periodicity.

Assuming web objects remain in a client's web browser cache for the entire trace period (26 days), the AP-coresidential client cache would attain an ideal hit ratio of 25%, which is less than the ideal hit ratio for same-client and same-AP caches within three minutes.

D. *Same-building and campus-wide repeated requests*

Same-building repeated requests are all the requests for which at sometime in the past there was another request for the same URL by a client from an AP in the same building as the one of the first request. We find that the fraction of repeated requests (i.e., hit ratio) varies from 75% to 15%.

We investigated how the number of HTTP requests and client population of a building may affect this hit ratio. For each building, the total number of unique clients that have sent at least one request from an AP in that building represents the *client population* at the building and the total number of requests sent from an AP in that building the *request demand*. The client population varies from 1,172 to 1 unique clients and the request demand ranges from

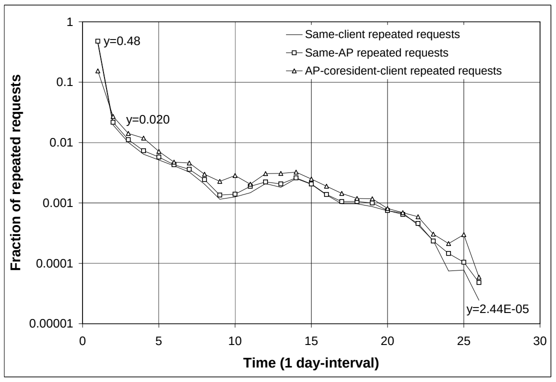

Fig. 2. Fraction of repeated requests within the entire trace. The number of requests considered is at least 7.6 million. Over 2,800 clients are represented.

1,929,399 to 5 requests. We sorted the buildings in decreasing order with respect to their client population and request demand. In both cases, there is a trend of declining hit ratio. However, the hit ratios across the buildings exhibit high variance and we cannot draw any strong conclusions. It is part of future work to investigate possible correlations of the hit ratios with building type (such as educational, administrative, residential), session type, and information access pattern.

We notice higher hit ratios than the ones reported in similar studies over the wired infrastructure. For example, in [9] the reported hits ratio for user populations of 200-1000 are between 45% and 52%, whereas our results for similar user population are in the range of 52% to 65%. This range corresponds to the six most populated buildings (191 to 1,172 clients). For the campus-wide repeated requests we count all the requests for which at sometime in the past there was another request for the same URL in our HTTP traces. Unlike these studies in which 25% to 40% of documents draw 70% of web access [3], our traces indicate that a 13% of unique URLs draws this number of web accesses. The UW study [9] reports a 59% ideal hit ratio for a similar user population size in the wired infrastructure in a university campus.

E. *Distinct-client location-dependent* URL*s*

The same-AP repeated requests and AP-coresident-client repeated requests give an indication of the spatial locality of a URL but do not capture entirely the dependency of repeated requests with a certain location. To better describe the dependency of a URL access to a location, we defined the distinct-client location-dependent property (DCLD) as follows: A URL exhibits DCLD in a building if a significant number of its repeated requests made by *different* wireless clients or in *different* buildings (or both) are made from that building.

Let us define the *popularity* α of a URL u to be $\sum_{i \in C^u} |B_i^u|$, where C^u is the set of distinct clients that have requested URL u and B_i^u the set of buildings from which client i has requested u. A URL u exhibits DCLD if there exists a building $b \in B$, such that:

$$\frac{\sum_{i \in C^u} In(b, B_i^u)}{\alpha} > T_{dcld},$$

where α is its popularity, T_{dcld} a certain threshold, B the set of all buildings, and $In(b, A)$ a binary function that indicates if $b \in A$. The higher the threshold is, the higher degree of spatial locality dependency a URL exhibits.

This definition disregards repeated requests from the *same* client made from the *same* building because we want to emphasize the sharing across different clients in a building. However, we do consider the requests issued by the same client across different buildings, since this indicates the client's continuous interest in the information.

Let us define the following terms: $N^{url}(\alpha)$ is the number of unique URLs with popularity great or equal to α. $N^{url}_{dcld}(\alpha)$ is the number of unique URLs that exhibit spatial locality dependency conditioned that their popularity is greater or equal to α. $N^{req}(\alpha)$ is the number of repeated requests in which we only count once the requests for the same URL by the same client and building, and conditioned to have popularity greater or equal to α. Similarly, we define the $N^{req}_{dcld}(\alpha)$ for requests for URLs with DCLD. We selected the DCLD threshold T_{dcld} to be .50. We computed the percentage of URLs N^{url}_{dcld} that exhibit DCLD with a popularity α in the range of 2 to 100. In our HTTP trace, 2,365,197 unique URLs occur for a total of 7,654,523 requests[1]. Out of the 7,654,523 there are $N^{req}(1)$ equal to 3,981,182 requests (i.e., sent by one or more clients or in different buildings or both in the HTTP trace).

Figure 3 shows the fraction of DCLD URLs as a function of the URL popularity α. It also displays the percentage of requests for these URLs. There are 290,130 unique URLs for which at least two clients or the same client from two different buildings tried to access them. Of these 290,130 URLs, 19% of them exhibit DCLD. As α increases, the fraction of DCLD URLs decreases logarithmically. Furthermore, DCLD URLs with high α truly exhibit spatial locality.

V. CLIENT ASSOCIATION PATTERNS

Although there are 6,186 active clients in our trace, we find that only 1,146 are active in an average day. Similarly, only 185 APs were active in an average day.

[1]Although there were 8,358,048 requests in the trace period, we were only able to establish the client and the AP involved in the request 7,654,523 times.

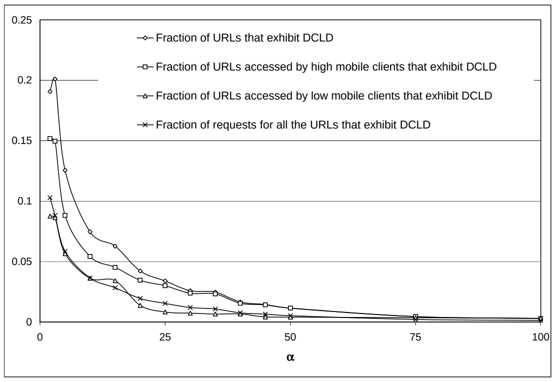

Fig. 3. Fraction of DCLD URLs and their respective requests. These measurements correspond to our entire HTTP trace of more than 7.6 million requests for over 2,800 clients.

Overall, APs in student residence buildings received more (re)associations than APs in academic and office buildings. In terms of busiest time of day, APs in academic and office buildings received more (re)associations during the day, with the peak around 1pm, whereas APs in student residence buildings received more (re)associations at night. These results are in agreement with those in [5]. We investigate the association patterns by examining how many and what type of clients, transitions, and sessions there are.

A. Client classification

We are interested in comparing the number of clients that there are each day, as well as finding out if they are roaming, mobile, or drop-in clients. We find that on an average day, there are 436 roaming, 231 drop-in, and 111 mobile clients. The number of daily roaming clients is about twice as much as those reported in [5], but the number of daily clients that visit two or more buildings is about the same.

We investigated and think that this could be due to the fact that the number of APs in our trace is about half the number of APs in [5], even though our user population is more than three and a half times larger. Our campus is larger than Dartmouth's, and therefore it is more likely that in Dartmouth two APs in two different buildings share a coverage area. This would explain the larger percentage of drop-in cards seen in [5].

A client's number of inter-building transitions is an indicator of its mobility. We obtain a relative measurement of its mobility by comparing this number to the total number of visits and inter-AP transitions. In our trace, we found the average of all clients to be 363 visits, 164 inter-AP transitions, and 32 inter-building transitions. The median of all clients is 57 visits, 40 inter-AP transitions and 6 inter-building transitions in the entire trace. If the average client visits an AP, the AP will be a different AP than the one it is currently connected to 28.8% of the time, and it will be in a different building 7.8% of the time. If the visit is to a different AP, then the likelihood that this AP is in a different building is 20.2%.

To model the mobility pattern of a wireless client, we would like to compute the characteristics of its movement while connected. For that we define the *AP path* to be the sequence of *continuous* inter-AP transitions. For example, if an wireless client that was originally disconnected, connects to APs 1, 2, 1, 1, and 10, before disconnecting, its path is "1 2 1 10". The length of this AP path is three. The *building path* is similarly defined. Figure 4 shows the mean and median for the maximum and mean AP and building path length of all users.

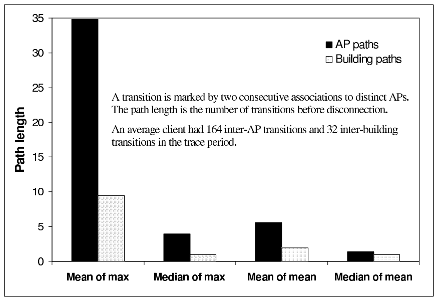

Fig. 4. Statistics for the path length of all 6,186 clients.

B. Session duration

We are also interested in investigating how long each client remains connected to the network. Our results show that most of the sessions, 55.1%, last less than 30 minutes, and 67.2% last less than one hour. We found that only 15.8% of our sessions lasted less than one minute (as opposed to 27% reported in [5]). We believe that since all of the incoming undergraduate students are required to buy a wireless laptop, there are more wireless clients that remain stationary in user's dorm. These stationary clients increase the number of long sessions, which could explain why our numbers are lower than those in [5] .

C. Next-state prediction

We measured the overhead for associating with an AP and found that the delay from the time the first association request was captured until the association is successfully completed has a 95% confidence interval for the mean of (136 ms, 157 ms). Such overhead in addition to end-to-end delays can be prohibitive for several real-time multimedia applications. The prediction of the next association can be used to mask this delay by buffering and prefetching data. APs can use it to predict their traffic and coordinate with their neighbor APs for load balancing and better utilization of their buffer and wireless bandwidth. In addition, the association protocol could be enhanced by advising the client to avoid hot spots.

We use a client's state history to develop a model that predicts the nth state of the client, given its most recent state history. Our prediction model is based on a Markov chain and uses the current state to predict the next. For each client, we construct the first order Markov chain based on the client's state history. Each state of the Markov chain corresponds to a *state* as defined in Section III. The transition probability from state j to state k is the relative frequency of the sequence of states $s_j s_k$ in the client's state history. This corresponds to the j, k entry of the transitional probability matrix $P^{(1)}$. We extend our prediction model by using the previous as well as the current state to predict the next. In this prediction model, we compute the relative frequencies of $s_i s_j s_k$. This corresponds to the i, j, k entry in the three-dimensional matrix $P^{(2)}$. [2]

We describe three variations on our prediction algorithms depending on the amount of history considered in building our two prediction models:

One-state history: This model is the one-state history as discussed above. The first $n-1$ states are used to build $P^{(1)}$. We predict the next state to be the state s_k such that k maximizes $P^{(1)}(j,k)$. The error making this single prediction of the next state n is $\varepsilon_n = 1 - P^{(1)}(j,k)$

One-state window: If the $n-1$th state occurs at time t, this model uses the sequence of states that occur between $t - 24$ hours and t to build its probability matrix. It then predicts the nth state in the same way the one-state history model predicts the next state.

Max of one-state window and history: This model computes the probabilities with which the one-state history model and the one-state window model predict the next state to be. It then selects the state that has the highest probability and predicts that state to be the next state with the same probability as that of the model chosen.

Two-state history : This model is the two-state history model discussed above. The first $n-1$ states are used to build $P^{(2)}$. Let the $n-2$th state be s_i and the $n-1$th state be s_j. We predict the next state to be the state s_k such that k maximizes $P^{(2)}(i,j,k)$. The error in the single prediction of the next state n is $\varepsilon_n = 1 - P^{(2)}(i,j,k)$.

Two-state window: If the $n-1$th state occurs at time t, this model uses the sequence of states that occur between $t - 24$ hours and t. It then predicts nth state in the same way the two-state history model predicts the next state.

Max of two-state window and history: This model compares the probabilities with which the two-state history model and the two-state window model predict the next state. It then chooses the state that has the highest probability and predicts that state to be the next state with the same probability as that of the model chosen.

We allow our models to "warm up" before making a prediction. Let $S_{training}$ be the collection of states used in the warm-up process, and $S_{predict}$ be the collection of states for which the models make a prediction. After obtaining all $s \in S_{training}$, each model predicts what the next state will be and then reads in the actual next state from $S_{predict}$. The prediction is then marked as correct or incorrect, and ε_n is computed. The model is then updated, and a new prediction is made. This cycle continues until there are no more states in $S_{predict}$ to be read in.

[2]There are some storage considerations. For example, a very mobile client can visit half of the total number of APs. Storing a single client's three-dimensional M matrix for 128 APs for a single day requires about 8MB of memory. Storing a four dimensional matrix would require about 1GB of memory.

To obtain an overall idea of how well the model is performing after a given number of predictions, we compute the correct prediction percentage and the prediction error $\bar{\varepsilon}$ thus far. For each model, the *correct prediction percentage* is the fraction of times that the next state was predicted correctly. The *prediction error* after predicting n states is defined as the mean of the error of all predictions made.

Only 3,984 (or 64% of) the 6,186 active clients have more than 25 syslog entries; the following results are for these clients only. A good rule of thumb is that if there are more than 30 samples, the central limit theorem can be applied. We find that there are only 31 clients with at least 8,012 states. The mean correct prediction percentages for predicting state 8,012 were 81.36%, 82.16%, and 84.85% for the one-state history, one-state window, and max of one-state window and history models, respectively. For the two-state history, two-state window, and max of both two-state window and history model, the mean correct prediction percentages were 83.68%, 83.19%, and 85.59%, respectively.

Figure 5 illustrates the percentage of correct predictions after each entry. The one-state history and the one-state window history model have similar correct prediction percentages, and that the two-state models perform slightly better than their one-state counterparts. The one-state history, one-state window, and max of one-state history and window had a prediction error of 0.26, 0.23, and 0.21, respectively, and their two-state counterparts had a prediction error of 0.23, 0.22, and 0.20, respectively. The standard deviations for the correct prediction percentages are all less than 0.19 for the one-state models and less than

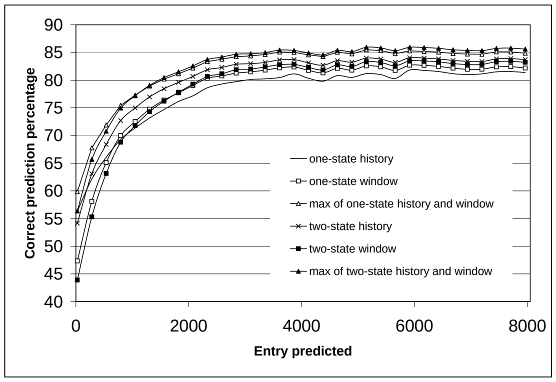

Fig. 5. Next-state prediction. Correct prediction percentage after each entry. Initially, there were over 3,900 clients. For the prediction of the last entry only 31 clients participated.

0.18 for the two-state models. The standard deviations for the prediction error are all less than 0.23 for the one-state models and less than 0.21 for the two-state models. Note that by maintaining information about the last 2,000 entries the max of two-state history and window achieves a correct prediction percentage of at least 82.17%. This suggests that if storage space is a concern, the model can be implemented in a slightly different manner that uses only a certain number of entries such that it is space efficient and still has a high correct prediction percentage.

We find that the top five percent of clients in terms of total number of inter-building transitions who also have 8,012 events have a correct prediction percentage of 79% for predicting state 8,012. Figure 6 illustrates the correct prediction percentage after 80,000 entries instead of 8,000 entries and Figure 7 shows the error in making each of those predictions.

We also incorporated a time component into the sequence of states as described in [2]. This method produces additional states by polling for a client's state at regular time intervals and thereby creates a state history based on both movement and time. Some clients are disconnected for long intervals of time, and during this time polling introduces long sequences of 0. This in turn tends to overestimate the performance of the predictor, since it will have an extremely high correct prediction percentage during periods of disconnection. We therefore decided to use this movement and time model, but only predicted the next state if the client's current state is a connected state. The mean correct prediction percentage using the max of two-state window and history model was 87% at the last entry for which there were more than 30 clients.

D. Revisits

We want to find how likely it is for a client to visit an AP that it has visited within a certain time interval. We may be able to improve the caching at an AP if we use information about the frequency that clients revisit that AP after visiting a different AP. This can provide some guidelines for how long a user's information (e.g., profile, cache) should be stored in an AP.

For a given client, we use its state history with a timestamp that indicates when the client visited each state. For a time interval W, we define its revisit probability at each state s_i as the fraction of times this client is visiting s_i within a time period W since its last visit to s_i , and also has at least one visit to any other state s_j in between the two visits to s_i. For a given AP, we compute the fraction of all visits made by all users that were revisits.

We found that the mean revisit probability for an one-hour interval W is 20% for clients and 40% for APs. However, the revisit probability varies drastically among APs and clients, varying between 0 and 95% among APs and 0 and 99% among clients. Figure 8 shows for each AP the probability that a visit at that AP was a revisit and Figure 9 shows the probability that a visit was a revisit for each client. In both figures the id of the APs and the clients were sorted in increasing order of revisit probability to improve readability. The median revisit probability

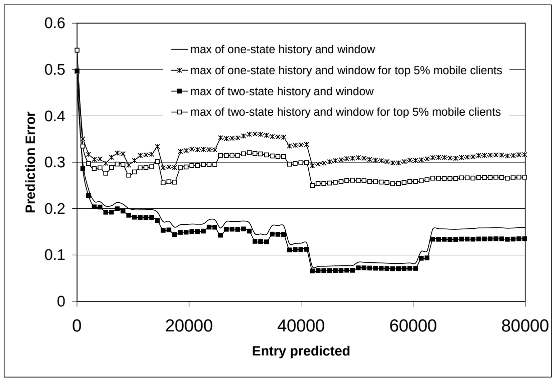

Fig. 7. Next-state prediction. Prediction error after each entry. Initially, there were over 3,900 total clients and 300 mobile clients. For the prediction of the last entry only 2 total clients and only 1 mobile client participated.

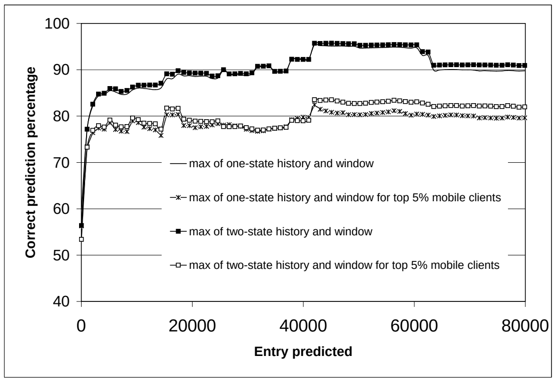

Fig. 6. Next-state prediction. Correct prediction percentage after each entry. Initially, there were over 3,900 total clients and 300 mobile clients. For the prediction of the last entry only 2 total clients and only 1 mobile client participated.

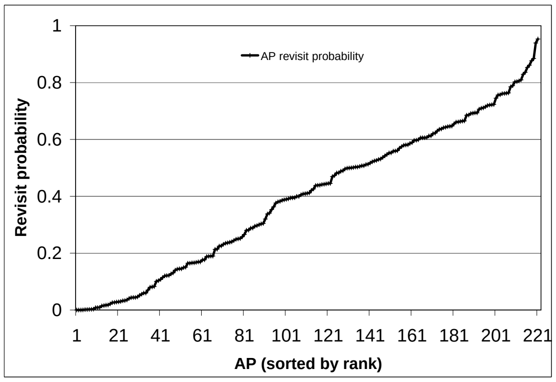

Fig. 8. Revisit probability for each AP.

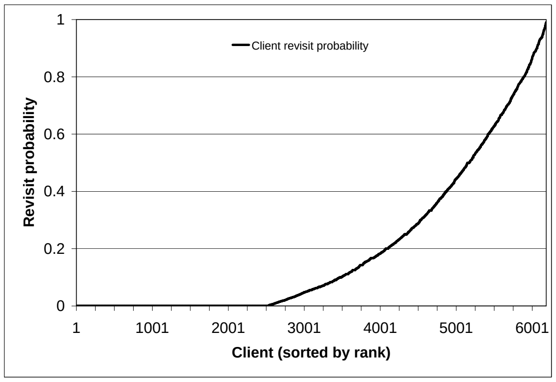

Fig. 9. Revisit probability for each client.

was 40% and 6% for APs and clients, respectively. Therefore, a cache with a lifetime of one hour at each AP would be beneficial.

E. Requests during transition periods

In this section, we identify the wireless information access that occurs during a transition from one AP to another. We define the *transition period* as the time interval that starts five seconds before a (re)association to a *new* AP and ends five seconds after that (re)association occurs. For example, if the client is associated with an AP AP_i and there is a (re)association with AP_j (where $i \neq j$) at time t, then the transition interval is $[t-5, t+5]$. A transition is an inter-building transition when the two APs are in different buildings. Tables III and IV show the access pattern during such transitions.

Access pattern (before and/or after AP transition)	**Clients**	**Requests**	**Requests per user** (mean, std dev)
any URL before or after	603	35,409	58.72, 162.82
any URL before and any after	289	1,144	3.95, 6.53
same URL before and after	100	1,039	10.39, 34.91

TABLE III
WEB ACCESS DURING WIRELESS TRANSITIONS BETWEEN APS.

Access pattern (before and/or after building transition)	**Clients**	**Requests**	**Requests per user** (mean, std dev)
any URL before or after	146	3,139	212.50, 43.45
any URL before and any URL after	40	90	2.25, 2.30
same URL before and after	9	12	1.33, 0.50

TABLE IV
WEB ACCESS DURING WIRELESS TRANSITIONS BETWEEN APS OF DIFFERENT BUILDINGS.

VI. CONCLUSIONS AND FUTURE WORK

To support wireless mobile users, it is crucial to provide robust and intelligent wireless infrastructures. This motivated us to explore characteristics of the wireless infrastructure that provide opportunities for caching, prefetching, wireless coverage planning, and resource reservation. We conducted a one-month measurement study of locality phenomena among mobile wireless users and their association patterns on a major university campus using the wireless infrastructure. This infrastructure provides coverage for nearly every building in the 729-acre campus.

We found that each client frequently requests objects that it has requested within the past hour, and occasionally requests objects that had been requested by other nearby users within the past hour. The overall ideal hit ratios of the user cache, cache attached to an AP, and peer-to-peer caching (where peers are coresident within an AP) paradigms are 51%, 55%, and 25%, respectively. A cache at each AP would achieve an ideal hit ratio of 55% for the whole trace, whereas a cache that serves the entire campus would achieve an ideal hit ratio of 71%. There are APs with higher ideal hit ratios; for example, one AP in an auditorium had an ideal hit ratio of 73% that corresponds to 40,064 requests (which is the total number of requests made by six distinct users). Same-AP caching is beneficial for these APs.

Unlike previous studies on the wired network in which 25% to 40% of documents draw 70% of web access [3], our traces indicate that 13% of unique URLs draws this number of web accesses. We plan to extend this study by comparing traffic characteristics (applications and content type) of the wired vs. wireless infrastructures at UNC for the same period. We are considering applying clustering techniques to detect user traffic and association patterns.

The peer-to-peer caching system demonstrates a 25% ideal hit ratio for web requests. However, the web is not primarily a location-dependent or collaborative application and as such we also observed a low percentage of URLs with the DCLD property. URLs accessed by high mobile users exhibit a DCLD percentage that is more than three and a half times that of the URLs accessed by low mobile users when α equals 25. The peer-to-peer caching systems that initially motivated this study, such as 7DS [7], require the objects to be cacheable. Stale objects should not be distributed, but many popular objects on the web are not cacheable by the HTTP standard [4]. It appears that content providers use cacheability to force reloads of their pages for reasons *other* than document freshness (such as distributing new advertisements). This use of the cacheability mechanisms works well enough in fully connected environments, but is a limiting factor for weakly connected systems as we describe here. We intend to measure cacheability of objects in wireless traffic and address this issue; ideally, an object should be cached only for its *true* useful lifetime, while content providers receive the feedback they need.

We are interested in learning how different traffic types correlate with buildings, user devices, and their locality properties, especially in the arena of location-dependent and collaborative applications for mobile users. For that, we are implementing several collaborative and location dependent systems, including a note-sharing system for use in presentations and also a location-sensing interactive map tool. We plan to perform measurements of the locality effects in settings with these applications deployed as well as corporate ones.

For the support of applications with strict real-time requirements, APs need to monitor user traffic and associations and apply efficient load balancing, admission control, and resource allocation mechanisms. Prediction algorithms for the traffic demand and visits per AP and client would assist the development of intelligent and robust wireless infrastructures. The Markov chain based predictions of the next state can achieve an 86% correct prediction percentage. We plan to expand the next-association prediction by incorporating additional information such as weekday, duration at an AP, different time-scale associations patterns of each user, and campus networking and physical topology.

We found that the median of revisits within one hour for all APs is 40%. The prediction of the client revisits to an AP within an interval of time and next association can be used to mask the end-to-end delay and association overhead. This can be done by buffering, prefetching, and maintaining data related to client interests, profile, or web access. APs can use similar techniques to predict their traffic, not only the number of associations, but also data transferred. They can use their predictions to coordinate with neighboring APs for load balancing and better utilization of their buffer and wireless bandwidth. In addition, the association protocol could be enhanced by advising the client to avoid hot spots and suggesting alternative APs.

We found the average of all clients to be about 363 visits, 164 inter-AP transitions, and 32 inter-building transitions during our trace period. If the average client visits an AP, the AP will be a different AP than the one it is currently connected to 28.8% of the time, and it will be in a different building 7.8% of the time. If the visit is to a different AP, then the likelihood that this AP is in a different building is 20.2%. We plan to extend this study and model the inter-building transitions and visit durations at an AP and in a building. Administrators can also use similar statistics as the ones performed in Section V to deploy efficient wireless infrastructures.

There are many challenges on how to provide efficient caching and prefetching mechanisms and content networks that enhance the information access of mobile users. To our knowledge, this is the first empirical study that analyzes spatial locality properties of the wireless web access. We believe that spatial locality can have a dominant impact on the mobile information access and this study sets the directions for exploring further such issues.

REFERENCES

[1] Anand Balachandran, Geoffrey Voelker, Paramvir Bahl, and Venkat Rangan. Characterizing user behavior and network performance in a public wireless lan. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, 2002.

[2] Amiya Bhattacharya and Sajal K. Das. LeZi-update: an information-theoretic approach to track mobile users in PCS networks. In *Proceedings of the Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 1–12, Seattle, Washington, USA, August 1999.

[3] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, March 1999.

[4] Bradley M. Duska, David Marwood, and Michael J. Feeley. The measured access characteristics of world-wide-web client proxy caches. In *USENIX Symposium on Internet Technologies and Systems, 1997*, Monterey, CA, December 1997.

[5] David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. Technical Report TR2002-432, Dept. of Computer Science, Dartmouth College, September 2002.

[6] Mark Lindsey. Correlations among nearby mobile web users. Master's thesis, Department of Computer Science, University of North Carolina at Chapel Hill, May 2003.

[7] Maria Papadopouli and Henning Schulzrinne. Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, Long Beach, California, October 2001.

[8] Diane Tang and Mary Baker. Analysis of a local-area wireless network. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 1–10, Boston, Massachusetts, USA, August 2000.

[9] Alec Wolman, Geoffrey M. Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M. Levy. On the scale and performance of cooperative web proxy caching. In *Proc. ACM Symposium on Operating Systems Principles*, Kiawah Island, SC, December 1999.