Supporting Diverse Mobile Applications with Client Profiles

Laura Bright
bright@cs.umd.edu

Samrat Bhattacharjee
bobby@cs.umd.edu

Louiqa Raschid
louiqa@umiacs.umd.edu

University of Maryland
College Park, MD 20742

ABSTRACT

We describe a best-effort scheme to support diverse applications in low-bandwidth mobile networks. The improvements due to our scheme are motivated by the observation that different applications typically have different preferences for latency and recency of data, which are not considered by traditional caching techniques. In our scheme, clients express their preferences using voluntary *profiles*: application-specific targets for latency and recency of data. We describe a complete framework for incorporating profile-based decision making into the cache utilization, downloading, and scheduling decisions at a mobile base station. We analyze the performance of profiles using simulations. Our results show that even simple profiles are useful in discriminating the service received by different applications, and using profiles in both caching and scheduling decisions improves performance. Further, clients who give uncooperative profiles increase the latencies of their own applications, thus providing all clients an incentive to provide fair and accurate profiles.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Miscellaneous

General Terms

Design, Performance

Keywords

mobile computing, caching, scheduling, service differentiation

1. INTRODUCTION

An increasing number of clients use mobile devices for diverse applications such as stock quotes, news headlines, travel information, weather reports, and email. As wireless connectivity becomes more pervasive, the number of mobile devices, applications, and services is likely to increase correspondingly. Currently most wide-area wireless networks have limited bandwidth (on the order of 10s [1] or 100s [30] of kilobits). As the number of clients and applications increase, it is necessary to better utilize the available bandwidth and improve QoS on low-bandwidth wireless networks.

There are two alternatives for supporting diverse mobile applications. One strategy is full QoS deployment, which will provide latency, bandwidth, and packet jitter guarantees, e.g., [24, 26]. However, such services require end to end deployment, and are unlikely to be widely available in the near future. A feasible alternative is to consider best-effort relative service discrimination schemes. These proportional service schemes do not provide absolute guarantees; however, they are locally deployable. The goal of this paper is to consider such a scheme and show that it is useful for real applications.

We present a best-effort scheme to intelligently arbitrate the shared resources in a mobile network using *client profiles*. A profile encapsulates a desired level of data recency and response latency, and can be specified on a per-client, per-application, or even per-request basis. This work is motivated by the observation that different clients may have different quality of service requirements for their different applications. For example, a client using a wireless device for instant messaging typically wants to receive data as quickly as possible, while the same client may tolerate some amount of latency while accessing e-mail. Profiles are designed to capture exactly these differing application-specific service requirements. We present a set of algorithms to implement such discrimination in mobile network cells.

We incorporate profiles into scheduling at the base station. Once a requested object has been retrieved by the base station (either from its cache or from a remote server), it must be scheduled for delivery to the mobile client. Scheduling is important in low-bandwidth environments since a single poor decision can affect many applications that get starved. The obvious technique is to use priority-based scheduling instead of FIFO scheduling. There has also been much work in on-demand scheduling for wireless data dissemination systems, e.g., [3, 6, 5, 29], with the goal of minimizing average latency. This work, and existing work in scheduling, does not consider client populations with different requirements or latency/recency tradeoffs. To best utilize the wireless bandwidth, we believe it is important to develop an integrated scheme that incorporates all of the application requirements, including data recency and latency of requests.

WoWMoM'02, September 28, 2002, Atlanta, Georgia, USA.

Profiles can leverage proxy caching near a base station. Since the requests of a number of mobile clients in a cell are multiplexed through the base station, maintaining a proxy cache at a server near the base station can leverage commonalities in client accesses and improve access latencies. The base station uses the profile of a request and knowledge of the system workload to make a decision whether to (a) serve an object from the cache or (b) download the object from a remote server. There has been much research in proxy caching for stationary clients, especially in the context of the WWW [16, 20, 23]. An obvious drawback to always using a proxy cache is that data becomes stale as data is updated at the remote servers. The design goal of some mobile applications, e.g., on-line trading applications, is to provide access to the most recent data. Some applications or users, however, may explicitly want to trade-off information recency for low access latencies, e.g., some clients may disconnect or leave their cell, and thus may be willing to accept stale information if it can be delivered to them quickly. Traditional caching schemes do not take such client requirements into account; however, these are exactly the situations where a profile-aware base station will be of use.

The main contribution of this paper is the design and evaluation of a service differentiation scheme using client profiles. We describe a complete framework for incorporating profile-based decision making into the cache utilization, downloading, and scheduling decisions at a mobile base station. Our profiles are simple (three parameters: a target recency, a target latency, and a priority), and are straightforward to implement and deploy at a wireless base station. We present a simulation-based evaluation of profiles. Our results show that simple profiles are effective at differentiating services in mobile environments. Further, we show that clients who provide inaccurate profiles hurt the latency of *all* requests, including their own applications. This gives all clients an incentive to provide fair and accurate profiles.

This paper is organized as follows: In Section 2 we describe the problem and our model in detail. In Section 3, we describe client profiles in detail and describe how to select different profile parameters. We present our simulation study in Section 4. We discuss related work in Section 5, and conclude in Section 6.

2. MODEL

We consider a set of mobile clients sharing the wireless bandwidth in a single cell. Clients send requests for objects to the base station in their cell. The base station is connected to a proxy cache as shown in Figure 1.

In this paper, we assume that base stations are equipped with functionality to make caching and scheduling decisions based on profiles. Another feasible alternative is to implement the profile specific functionality at a host colocated with the base station. We do not consider such implementation issues further in this paper; instead, we use the generic term base station to refer to the entity with the caching and scheduling functionality.

When an object is requested, the base station first decides whether to use a cached copy of the object or to contact the remote server to download a fresh copy. When an object becomes available at the base station, it is scheduled for delivery to the client via the wireless downlink. The base station transmits objects in the order defined by a scheduling algorithm.

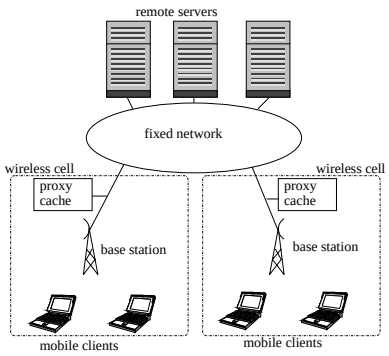

Figure 1: System Model

Application	Cacheable?	Low Lat?	Low Age?	Examples
1	Y	Y	Y	stock quotes
2	Y	N	N	news, weather
3	N	Y	N/A	instant messaging
4	N	N	N/A	email

Table 1: Categories of Client Applications and Profiles for non handoff

2.1 Applications and Profiles

In our model, the available wireless bandwidth is shared by multiple mobile clients subscribing to different applications. We characterize applications by the importance of latency and recency. We define *latency* as time elapsed from the time a request arrives at the base station to the time the last bit of the object is delivered to the client. We use *age* as our recency metric. *Age* is the estimated number of times an object has been updated at the remote server since it was cached as defined in [22]. An object with $age = 0$ is considered *fresh*, and an object with $age > 0$ is considered *stale*.

Profiles can be set by either clients or service providers, as described in Section 3.1. We consider the following categories (summarized in Table 1) of client applications.

1. Applications that generate cacheable requests, and prefer low latency and fresh data, e.g., stock quotes.
2. Applications that generate cacheable requests, and can tolerate higher latency for fresh data but may potentially accept stale data if it can be delivered quickly, e.g., news, weather.
3. Applications that generate non-cacheable messages and require low latency, e.g., instant messaging.
4. Applications that generate non-cacheable requests but can tolerate higher latencies, e.g., e-mail.

Clearly, different clients or providers may map individual applications into very different categories. This is accomodated by profiles since clients (or providers) themselves map applications to categories. Note that in the classes of applications we have described, it suffices to have only two levels of latency discrimination; interactive applications want responses as fast as possible while the other applications can tolerate delay. We note that it is possible to define other classes of applications.

2.2 Assumptions and Restrictions

We make the following assumptions:

- Clients share the available wireless downlink bandwidth, and objects are unicast from the base station to the clients. This is how cellular providers currently serve mobile clients on a data channel.
- We assume that every object has a Time-to-Live (TTL) [20] assigned by a remote server. A TTL is an estimate of how long a cached object will remain fresh. We do not consider the effects of inaccurate TTL estimates on our results. Our implementations of the different downloading polices rely on the same estimates of the update frequencies of objects; therefore the effects of inaccurate TTL estimates on all policies are comparable.
- We assume that the time to read an object from the cache is negligible; therefore all cached objects are available to be sent to clients immediately. This assumption is reasonable because latency on a wide area fixed network is typically much higher than the latency of accessing a local cache.
- We do not consider the cost of sending requests on the wireless uplink, as the sizes of these requests are typically much smaller than the objects transmitted on the downlink.

3. PROFILE-DRIVEN ACCESS

Profiles allow clients to differentiate the service provided to them and to identify their target requirements. We first describe potential deployment scenarios and granularities of profiles. We then describe the profile parameters, and how to choose values for them. Finally, we describe the details of the profile-driven downloading and scheduling at the base station.

3.1 Profile Deployment and Granularity

In this paper we assume that clients set their own profiles and communicate them to the base station. Clients can map either applications or individual requests to specific profiles. We note that it is also possible for service providers to set profiles. In this case, the service provider could map profiles to either clients or applications.

If clients set their profiles, parameter values can be appended to requests and passed to the base station, assuming that profiles can be encoded suitably succinctly. Thus, the base station does not need to store any profile specific information, and the profiles themselves do not add extra storage or retrieval overhead at the base station. Also, clients can set and tune their profiles locally, without the overhead of communicating with the base station.

We note that clients can distinguish between different applications of the same request type, e.g. HTTP requests. For example, both stock quotes and news headlines are typically requested by making HTTP requests from a web browser. These applications can be assigned different profiles by mapping different domain names to different profiles. For example, a client could specify that requests to `finance.yahoo.com` require the most recent data, but all other web requests can tolerate stale data.

3.2 Profile Parameters

Our profiles include three parameters: a target latency T_L, a *priority*, and a target age T_A. We describe how to choose these parameters for specific applications.

3.2.1 Choosing Priority

The first parameter is a *priority* which is used to schedule objects that are available for delivery on the wireless downlink. We use a simple 1-bit priority scheme, which is sufficient for the set of applications (IM, email, web browsing) we have considered in this paper. We specify the *priority* for applications as either *LowPriority* (e.g., casual web browsing, email) or *HighPriority* (e.g., instant messaging). Intuitively, *HighPriority* should be used for requests where it is important to deliver data as quickly as possible, and *LowPriority* should be used for requests that can tolerate higher latencies if necessary. We use two values, *LowLatency* and *HighLatency*, to map these priorities to deadlines, and then schedule objects for delivery using an EDF scheduler. This scheme assures that *LowPriority* requests will not get starved.

Since the access bandwidth of different wireless networks can vary widely, we take both the wireless bandwidth and fixed network latencies into account when computing *LowLatency* and *HighLatency*. We compute the value of *LowLatency* as the sum of the median fixed network latency and the downlink latency. We compute the value for *HighLatency* as:

$HighLatency = C \times LowLatency$

where C is a constant $\geq$1.0. Note that when $C = 1$, *LowPriority* and *HighPriority* requests have identical priorities and are processed in a first come, first served manner.

3.2.2 Choosing Target Latency

Clients assign a target latency T_L to each request. T_L indicates how long a client is willing to wait for fresh data, and is used to determine when to download an object and when to use a cached copy. These T_L are chosen to be either *LowLatency* or *HighLatency*. This parameter T_L is used at the base station to determine when to download an object and when to use a cached copy. It differs from *priority* which is used for scheduling objects that are available for delivery to clients.

3.2.3 Choosing Recency

In addition to specifying a target latency T_L, clients specify a target age T_A. Recall that age is defined as the number of times an object has been updated at the remote server since it was cached. Clients who require the most recent data set T_A to 0. In an implementation, target age would be expressed as a time (instead of number of revisions). It is relatively straightforward to convert a last modified time to a number of revisions as long as appropriate information about revision history is available or can be estimated [15].

3.3 Using Profiles at the Base Station

We describe profile-driven decision making at the base station. We first decribe how profiles can be communicated to the base station. We then describe how the base station makes the decision to download an object or use a cached copy, and how it schedules objects for delivery.

3.3.1 Configuring and Communicating Profiles

As described in Section 3.1, profiles can be specified at a granularity determined by the client and/or service provider. Clients can communicate their profiles to the base station assuming the profile parameters can be encoded succinctly, for example by including them in a request's HTTP header. This eliminates the overhead of storing profiles at the base station and gives greater flexibility to the client.

3.3.2 Profile-Driven Downloading

When an object is requested, the base station first makes a decision as to whether to use a cached copy of the object or contact the remote server to download a fresh copy. It makes this decision using a scoring function which assigns a score of 1 to any value that meets or exceeds the target value (T_A or T_L), and approaches 0 as the actual value gets further from the target. Intuitively, the higher the score of an object, the better it meets the targets specified in a client's profile. We present one function below; we note that other functions with similar properties could also be used.

We define *Age* to be the estimated age of a cached object, and define *Latency* to be the estimated latency of downloading the object. We define our function *Score* as:

$$Score(T, x) = \begin{cases} 1 & \text{if x} \leq \text{T} \\ 1/(x - T + 1) & \text{otherwise} \end{cases}$$

Where T is the target value (T_A or T_L) selected by the client and x is the estimated value (*Age* or *Latency*). This function assigns a score of 1 to any value less than or equal to the target, and approaches 0 as the values get further away from the target. To estimate the latency of downloading an object, we use the average latency of previous requests for that object as in [4]. We estimate the age of a cached object using its last known modification and its TTL estimate[13].

3.3.3 Combined Decision Function

We define a combined decision function as the average of Score(T_L, *Latency*) and Score(T_A, *Age*). Note that a weighted average can also be used to better tune the function and provide upper bounds[10]; for simplicity we omit the details from this paper. When an object is requested, we compute the score of either downloading the object (*D_Score*) or using the cached copy (*C_Score*). The profile-driven download policy is as follows: *When an object is requested, if* D_Score > C_Score*, the object is downloaded from the remote server. Otherwise the cached copy is used.*

We define *Svc(queue)* as the expected amount of time it will take to serve outstanding requests with higher priority on the wireless downlink. We use this information, combined with the object's size and downlink bandwidth, to estimate the transmission time of an object. Since downloading always provides the most recent data, *Score(T_A, Age)* is always 1.0. Therefore, we compute *D_Score* as follows:

$$D\_Score = \tfrac{1}{2} * 1.0 + \tfrac{1}{2} Score(T_L, \max(Svc(q), Lat))$$

We estimate the latency to be the maximum of the service time of objects remaining in the queue ahead of this object and the fixed network latency of downloading the object.

We compute *C_Score* as:

$$C\_Score = \tfrac{1}{2} Score(T_A, Age) + \tfrac{1}{2} Score(T_L, Svc(q))$$

PROFILE-DRIVEN DATA ACCESS(OBJECT *obj*, PROFILE *P*)

```
if obj is in cache
    use P to compute C_Score and D_Score
    if D_Score > C_Score
        Request obj from remote server
        When obj arrives:
            ENQUEUE_EDF(obj, ServiceQueue)
            Refresh obj in cache
    else
        ENQUEUE_EDF(obj, ServiceQueue)
    end if
else      { obj is not in cache }
    Request obj from remote server
    When obj arrives:
        ENQUEUE_EDF(obj, ServiceQueue)
        Insert obj in cache
        if cache is full
            replace using LRU
        end if
end if
```

Figure 2: Combined Profile-Driven Algorithm

Note that when *Svc(queue)* is greater than the fixed network latency, *D_Score*>*C_Score*. This means that when the downlink is congested, more recent data can be downloaded in response to client requests because clients have to wait for their data anyway.

3.3.4 Profile-Driven Scheduling

When a requested object becomes available at the base station, it is added to the service queue and is scheduled to be sent to the client via the wireless downlink. In our profile-driven scheme, the base station transmits objects sequentially using EDF scheduling, where the priorities are either *LowPriority* or *HighPriority* as described in Section 3.2.

All of the components described above are integrated into a single algorithm at the base station. This algorithm is shown in Figure 2.

4. SIMULATION RESULTS

Our simulation results show that using profiles can improve performance. We first describe our simulation environment and parameters. We then present our results. Our key results are as follows:

- Profiles are effective at differentiating service of different applications.
- Effective use of the proxy cache can improve the latency of requests, even when there is contention on the wireless downlink.
- Too many *HighPriority* requests increase the latency of *all* requests. This gives clients or service providers an incentive to differentiate applications into *HighPriority* and *LowPriority*, and ensure that there are not too many *HighPriority* applications.

4.1 Simulation Model and Environment

We modeled a wireless downlink of 128 kbps, which is representative of emerging Third Generation (3G) [30] wireless networks. This is a dedicated data channel shared by multiple clients at a base station. Client requests were uniformly distributed between the 4 applications shown in Table 3, with about 25% of requests for each type of application. We discuss the effects of varying distributions in Section 4.2.2.

4.1.1 Parameters

We considered a world of 100,000 objects. We used the following parameters in our simulation; they are summarized in Table 2.

- *Object Size and Popularity*: We ran experiments with both variable and uniform object sizes. For simplicity, we report on the results for the uniform object size; results for variable sizes were comparable. We considered objects with size 12.8 kbits. These are representative of data that web sites currently provide to wireless web clients [14].

  We skewed object popularity using a Zipf-like distribution. Work reported in [9] showed that web accesses typically follow a Zipf-like distribution, where the ith most popular object has popularity proportional to $1/i^{\theta}$, where θ is a value between 0 and 1.0. We generated a distribution with θ= 0.7, which was typical of the web traces analyzed in [9].

- *Request Arrival Rate*: We generated requests with exponential interarrival times, and considered mean request arrival rates between 0 - 10 requests per second. This corresponds to 0 - 100% utilization on the wireless downlink.

- *Fixed Network Latency*: This is the estimated time to download an object from a server on the fixed network. To model fixed network latencies, we used trace data from NLANR [18]. The data was gathered from client-side web proxy caches at several sites on June 27, 2001 and consisted of approximately 1.3 million requests. To reduce the effects of network and server errors we considered only requests with latencies of less than 5000 msec. The distribution of these values was highly skewed, with a median of approximately 200 msec and a mean of approximately 500 msec. 90% of the requests had latencies less than 1400 msec.

- *Priority*: We considered two different priority values, *HighPriority* and *LowPriority*. The corresponding latency values are shown in Table 2.

- *Update Rate*: This is the frequency with which an object is updated. In our simulation this value was uniformly distributed in the range of once every 10 minutes to once every 2 hours.

- *Cache Size*: We considered a default cache size of 160 MB (about 10% of the world size).

4.1.2 Setup

We implemented a request-level simulation environment in C++. We ran all simulations on a Sun Sparc 20 workstation running Solaris 2.6. We assumed an initially empty cache. When the cache was full, we evicted objects using the Least Recently Used (LRU) policy. To keep cached objects up to date, we refreshed objects in the background at a rate inversely proportional to the workloads. This ensured that even at low workloads, the cache was reasonably fresh and increased the number of requests that could be served from the cache. The cache was refreshed in a fixed order in a round-robin manner; this strategy was shown to be near-optimal in [15].

We ran each simulation for 40000 requests to warm up the cache, then ran the simulation for an additional 80000 requests to collect measurements. We repeated each experiment 10 times and ran 95% confidence intervals. These intervals were very small in all cases and we do not show them in our plots.

The settings of the profiles are shown in Table 3. Note that for the uncacheable requests, there is no need to set values for T_A and T_L, because these objects will always be downloaded.

Parameter	Range
Request Rate	0 - 10 requests/sec
Downlink Bandwidth	128 kbps
Object Size	12.8 Kbits
Downlink Latency	100 msec
Workload	0 - 10 requests/sec
Update Rate	10 min - 2 hrs
Cache Size	160 MB
Fixed Netwk Latency	0 - 5000 msec
Median Fixed Netwk Lat	200 msec
LowLatency	300 msec
HighLatency	900 msec

Table 2: Simulation Parameters

Application	T_A	T_L	priority
1 (stock)	0	HighLat	HighPriority
2 (news)	2	LowLat	LowPriority
3 (IM)	N/A	N/A	HighPriority
4 (email)	N/A	N/A	LowPriority

Table 3: Application Profiles

4.1.3 Algorithms and Metrics

We compare *Profile*, our profile-driven downloading approach combined with EDF scheduling, against a profile-unaware scheme (*No Profiles*), which uses traditional TTL[13, 20] cache consistency and FIFO scheduling. Recall that TTL assigns each object an estimated Time-to-Live. Some servers provide this estimate; when it is not provided, it can be estimated using techniques decribed in [13, 20]. For the No Profiles approach, if a requested object is in the cache, it is delivered to clients only if its TTL has not expired. Otherwise a fresh copy is downloaded from the remote server.

We report on the average latency of requests. We define latency as the time elapsed from the time a request arrives to the time the last bit is delivered to the client.

4.2 Results

4.2.1 Effect of Using Profiles

Our first set of results compares the latency of the two non-cacheable applications (IM and Email) both with and

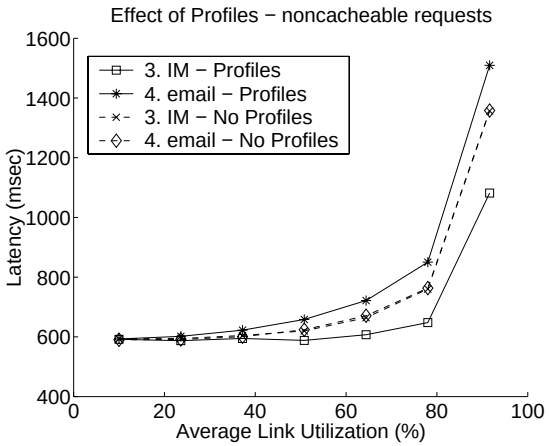

Figure 3: Effect of Using Profiles

without profiles. Recall that when profiles are not used, objects are scheduled for delivery using FIFO scheduling. The results are plotted in Figure 3. The key observation is that Profile effectively differentiates services for the two applications. Under light workloads, there is little contention for the wireless downlink and all applications have the same average latency regardless of the use of profiles. However, as the workload increases, the latency of both the Email and the IM requests increase at the same rate when FIFO scheduling is used. In contrast, Profile trades off the email response time to maintain a low latency for IM requests. The important observation is that at up to 80% utilization, the latency of the IM requests does not increase significantly. When the utilization increases beyond 80%, even Profile cannot maintain constant response time, but the relative gain from using profiles *increases*.

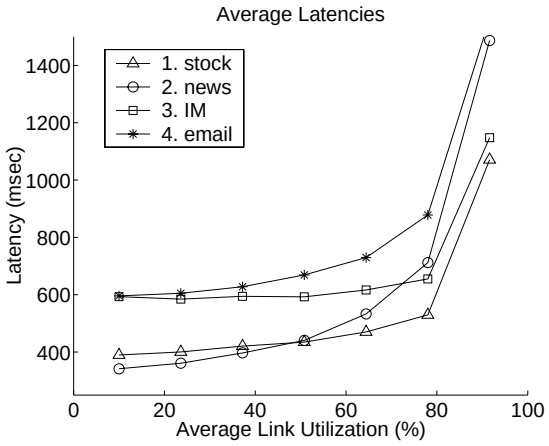

Figure 4: Average Latencies for Different Applications

We now consider how Profile differentiates services for all four applications. Figure 4 plots the latencies for all applications as a function of utilization on the downlink. The first observation is that at less than 50% utilization, the cacheable applications (stock and news) have lower latencies than the non-cacheable applications (IM and Email) because the cache reduces the fixed network latencies. Thus, under lighter workloads, caching data at the base station can significantly reduce the latencies of cacheable objects. A second observation is that using cached data improves the latencies of the cacheable applications (stock and news), independent of priority. At less than 50% utilization, the news requests have a slightly lower average latency than the stock requests. This is because the news application can tolerate stale cached data, while the stock requests require the most recent data.

As the workload increases, however, Profile is able to maintain the low latency of stock requests due to their *HighPriority* deadline. In contrast, the latency of the news requests increases. Thus, Profile can differentiate services and keep the latencies of the *HighPriority* stock requests relatively constant while the latencies of the *LowPriority* requests increase. At up to 80% utilization, both cacheable applications (stock and news) have lower latencies than both non-cacheable applications (IM and email). Thus, caching is beneficial even for *LowPriority* applications.

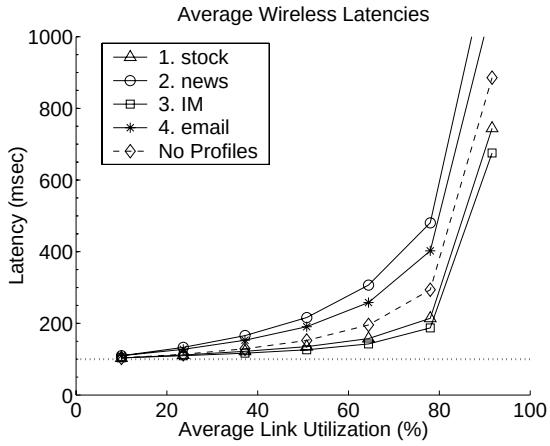

Figure 5: Average Wireless Downlink Latencies for Different Applications Using Profiles

We now consider the benefits of using Profile priorities in EDF scheduling. Figure 5 plots the wireless downlink latencies for all four applications, i.e. the amount of time to deliver data to clients *after* the data becomes available at the base station. For comparison, we plot the latency of No Profiles (TTL cache consistency and FIFO scheduling) for all requests. Recall that we use a fixed file size of 12.8 kbits and a downlink bandwidth of 128 kbps, so the minimum latency on the wireless downlink is 100 msec, shown by the horizontal dotted line. The key observation is that the *HighPriority* requests (stock and IM) have latencies within 25% of 100msec at up to 50% utilization. Thus, scheduling delay at the wireless base station has a minimal effect on the latency of these requests. We conclude that under reasonable workloads, EDF scheduling at the base station can control the wireless downlink latencies of different applications.

4.2.2 *Effect of Percentage of* HighPriority *requests*

We have assumed that all clients are cooperative and assign *LowPriority* to applications such as email and news that

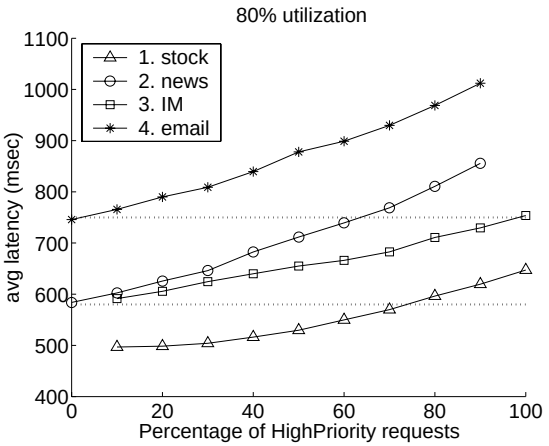

Figure 6: Effect of the Percentage of *HighPriority* Requests on Latencies

can tolerate higher latencies. We now consider the effects of the number of *HighPriority* requests on the performance of the system. Our results show the need for either clients or service providers to assign different priorities to different applications to fully exploit the benefits of profiles.

We vary the percentage of *HighPriority* requests from 0-100%, and plot the latencies of all four applications when the downlink is at 80% utilization in Figure 6. For a given percentage of *HighPriority* (or *LowPriority*) requests, there was an equal number of cacheable and non-cacheable requests. For example, when 40% of requests were *HighPriority*, this corresponds to 20% IM, 20% stock, and the 60% *LowPriority* requests were 30% email, and 30% news.

We plot the results in Figure 6. As the percentage of *HighPriority* requests increases, the latency of all applications increases. We first compare the performance of the non cacheable applications (IM and Email). When all requests are *LowPriority* (i.e., 0% *HighPriority*), the latency of Email is 750 msec. When 0 - 40% of requests are *HighPriority*, the average latency of the IM requests is in the range 600-650 msec. As the percentage of *HighPriority* requests increases, the latency of the IM requests reaches the lowest latency of email.

We observe similar behavior for the cacheable requests. Consider the lowest latency of the *LowPriority* news requests. When all requests are *LowPriority*, the latency is 600 msec. If we consider the latency of the *HighPriority* stock requests, as the percentage of *HighPriority* requests increases, the latency of the stock requests exceeds 600 msec.

These results indicate that when there is a high percentage of *HighPriority* requests, profiles can no longer provide service differentiation and meet application requirements. This is because profiles trade off latencies of *LowPriority* requests to better serve *HighPriority* requests. This implies that in a network with only *HighPriority* requests, a relative service differentiation scheme such as profiles is not useful and some other mechanism such as per flow scheduling is required.

5. RELATED WORK

There is much work in caching in mobile environments that is focused on client-side caching strategies to make efficient use of the wireless bandwidth, e.g. [8, 2]. WebExpress [21] is a system that aims to reduce the latency of Web access in mobile environments. The focus of this work is on reducing wireless traffic volume and protocol overheads.

There is also relevant work in web cache coherency e. g. [16, 20]. One approach is to assign each object in the cache a time-to-live (TTL) [13, 20], either using heuristics or simply using a TTL value assigned by a server. However, TTL requires validating expired objects, which may increase latency. Work in [7] allows cached values to deviate from server values in a controlled way; however, this requires cooperation from servers and may not scale to a large number of clients.

Support for diverse applications has been considered for both fixed networks[12, 19, 28, 25] and mobile networks[24, 27]. This work shares our goal of provisioning limited resources according to the needs of applications, but the emphasis is on providing QoS guarantees.

Work in broadcast scheduling [2, 29, 3, 6, 5] aims to minimize the average latency of client requests. Work in [6] considers the problem of data staging, i.e. bringing requested objects into main memory so they can be broadcast to clients. However, this work does not consider updates to cached objects. Lastly, note that the service differentiation scheme described in this paper is similar in spirit to the relative service differentation scheme described in [17]. However, we specifically take cache and mobility issues into account, and do not provide the same strict set of guarantees as in [17].

6. CONCLUSIONS

We have presented a scheme to improve data access for clients using diverse applications in mobile environments. Our scheme leverages client *profiles*, which allow clients to specify targets for latency and recency of data. The main contribution of this work is the definition of a set of intuitive application profiles and mapping them to real application scenarios. We believe our scheme is simple enough to be implemented at base stations, and we have presented algorithms for incorporating profiles with the caching, downloading, and scheduling algorithm at the base station. It is possible to incrementally deploy profiles, and we have outlined two different deployment scenarios. Our results show that profile-based access benefits application designers and clients. As the number of mobile clients grow and more applications share the same network, we expect the benefits from a service discrimination scheme like profiles to increase. We are currently implementing a testbed for profiles using the Squid Proxy Cache[11], and plan extensive analysis to further evaluate the performance of profiles.

7. REFERENCES

[1] Cellular digital packet data system specification: release 1.1. *CDPD Forum*, January 1995.

[2] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: Data management for asymmetric communication environments. *Proc. ACM SIGMOD Conference*, 1995.

[3] S. Acharya and S. Muthukrishnan. Scheduling on-demand broadcasts: New metrics and algorithms. *Proc. MOBICOM*, 1998.

[4] S. Adali, K.S. Candan, Y. Papakonstantinou, and V.S. Subrahmanian. Query caching and optimization in distributed mediator systems. *Proc. ACM SIGMOD Conf.*, 1996.

[5] D. Aksoy and M. Franklin. Scheduling for large-scale on-demand data broadcasting. *Proc. IEEE INFOCOM Conf.*, 1998.

[6] D. Aksoy, M. Franklin, and S. Zdonik. Data staging for on-demand broadcast. *Proc. Very Large Data Bases (VLDB)*, 2001.

[7] R. Alonso, D. Barbara, and H. Garcia-Molina. Data caching issues in an information retrieval system. *ACM. TODS Vol. 15, no. 3*, 1990.

[8] D. Barbara and T. Imielinski. Sleepers and workaholics: Caching strategies in mobile environments (extended version). *VLDB Journal Special Issue of the best of SIGMOD System-Oriented Papers*, 1995.

[9] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. *Proc. IEEE INFOCOM*, 1999.

[10] L. Bright and L. Raschid. Using latency-recency profiles for data delivery on the web. *Proc. Very Large Data Bases*, 2002.

[11] Squid Proxy Cache. *http://www.squid-cache.org*.

[12] Z. Cao and E. Zegura. Utility max-min: An application-oriented bandwidth allocation scheme. *Proc. INFOCOM*, 1999.

[13] V. Cate. Alex - a global filesystem. *Proc. USENIX File System Workshop*, 1992.

[14] Cellmania. *http://www.cellmania.com*.

[15] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. *Proc. ACM SIGMOD Conf.*, 2000.

[16] E. Cohen, B. Krishnamurthy, and J. Rexford. Improving end-to-end performance of the web using server volumes and proxy filters. *Proc. SIGCOMM*, 1998.

[17] C. Dovrolis and P. Ramanathan. A case for relative differentiated services and the proportional differentiation model. *IEEE Network*, 1999.

[18] National Laboratory for Applied Network Research. *ftp://ircache.nlanr.net/Traces*.

[19] P. Goyal, H. Vin, and H. Cheng. Start-time fair queuing: A scheduling algorithm for integrated services packet switching networks. *Proc. SIGCOMM*, 1996.

[20] J. Gwertzman and M. Seltzer. World wide web cache consistency. *Proc. USENIX Technical Conference*, 1996.

[21] B. Housel and D. Lindquist. Webexpress: A system for optimizing web browsing in a wireless environment. *Proc. MOBICOM*, 1996.

[22] Y. Huang, R. Sloan, and O. Wolfson. Divergence caching in client-server architectures. *Proc. PDIS*, 1994.

[23] T. Kroeger, D. Long, and J. Mogul. Exploring the bounds of web latency reduction from caching and prefetching. *Proc. USENIX Symposium on Internet Technologies and Systems*, 1997.

[24] K. Lee. Adaptive network support for mobile multimedia. *Proc. MOBICOM*, 1995.

[25] R. Liao and A. Campbell. A utility-based approach for quantitative adaptation in wireless packet networks. *Wireless Networks*, 7(5):541–557, 2001.

[26] S. Lu, V. Bharghavan, and R. Srikant. Fair scheduling in wireless packet networks. *IEEE/ACM Transactions on Networking*, August 1999.

[27] B. Noble and M. Satyanarayanan. Experience with adaptive mobile applications in odyssey. *Mobile Networks and Applications*, 4, 1999.

[28] I. Stoica, H. Zhang, and T.S.E. Ng. A hierarchical fair service curve algorithm for link-sharing, real-time and priority services. *Proc. SIGCOMM*, 1997.

[29] C. Su and L. Tassiulas. Broadcast scheduling for information distribution. *Proc. IEEE INFOCOM*, 1997.

[30] Third Generation Wireless Systems. *http://www.fcc.gov/3G*.