Preparing the Edge of the Network for Pervasive Content Delivery

***Daby M. Sow*[1], *Guruduth Banavar*[1], *John S. Davis II*[1], *Jeremy Sussman*[1], *Mugizi R. Rwebangira*[2]**

[1] IBM T. J. Watson Research Center
30 Saw Mill River Road,
Hawthorne, NY, 10532 - USA
{sowdaby,banavar,davisjs,jsussman}@us.ibm.com

[2]Systems and Computer Science Department
Howard University
Washington DC, 20059 - USA
mrweba@scs.howard.edu

Abstract

In this paper, we address the problem of delivering content in a pervasive environment characterized by high variability in network conditions, client devices and user context (e.g. location and preferences). This variability results in non-uniform user experience due to high and variable latency and could lead to user frustration during service access. We propose a methodology to tackle this issue and present a test-bed system that we are currently developing to verify our hypotheses.

1. Introduction

Pervasive computing promises an environment in which people will be able to access computation and information anywhere at any time. Networked embedded systems and widespread wireless access will facilitate ubiquitous connectedness. One of the main factors differentiating pervasive applications from other applications is the inherent variability within pervasive environments. There are three factors that contribute to this variability:

- *Device Heterogeneity*. Client devices have different modalities and use different presentation formats. Hence, the same content is often presented differently on different devices.
- *Network Infrastructure:* There is large variation in the physical characteristics of wireless channels and this affects the performance perceived by the end user. This is due not only to the number of different such technologies available today but also to inherent properties of wireless channels like multi-path fading problems, distance between client and base stations, and interference problems resulting from shared spectrum.
- *User Context*: Services available to the user may change over time and with the user's location. For example, services accessed in a professional environment are very different from the ones accessed in a home environment.

To achieve the true goal of pervasive computing – that computation and information be accessible from any device at any time in any place – a basic level of application usability must be attained. The primary problem resulting from the factors of environment variability listed above is a potential reduction in user satisfaction due to high and variable latency. Due to heterogeneity of pervasive devices, it is necessary to have format transformation (or transcoding) capabilities in content delivery networks, and such transcoding operations introduces latency that will be perceived by the user. Depending on the user's location, the available link technologies and the number of active connections operating in the same frequency band, the user experience while accessing services changes dramatically. This experience changes also with the context of each user. For example, discovering and binding to the appropriate services at each location introduces additional latency. If users are all in the same context and access the same content, then today's caching systems could limit the latency variations. But in general, current state of the art content delivery mechanisms do not solve this variability problem.

The ideal solution to providing uniform low latency is to load client devices with customized applications. If client devices do not have the ability to store several such applications, the alternative is to prepare the edge of the network to always be ready to deliver the application as soon as the mobile user requests it (and live with the network infrastructure variability issue). If all of the content needed by the user were available in the correct format at the access point closest to the user at all times, the perceived latency would be minimal. Unfortunately, this ideal scenario is impossible to realize, since it implies full replication of all applications and services, on all client devices, or on all client proxies at the edge of the network.

The impossibility of the above solution forces us to reformulate our scenario into a challenging optimization problem involving limited client and network resources. To overcome the limitations, we propose the addition of *a layer of pre-fetching middleware* that gives the illusion of full replication by intelligently predicting the usage patterns of end users and caching only those applications and services needed to prepare the environment appropriately. Building such a system is a difficult task. A useful solution requires an understanding of application offloading to the edge of the network, cache invalidation, the interaction of the cache with application customization, and the security and privacy of both the personal information and the usage patterns of the client.

Within a pervasive computing environment we believe that the efficiency of pre-fetching can be improved by leveraging a rich set of meta-information. This meta-information is associated with both usage patterns of end users and the back-end applications being accessed. Our work is designed to investigate how much improvement these approaches will provide over traditional content delivery systems.

In the remainder of this paper, we present background information followed by a description of the two sets of meta-information, namely usage patterns of end users and the structure of back-end applications being accessed. We discuss how these pieces of extra information can be used to make pre-fetching efficient and hence to provide solutions to the latency problem of pervasive computing. Finally, we describe a test-bed that we are currently building to validate our assumptions and test our hypotheses.

2. Background

The use of pre-fetching originated in computer architecture as instruction fetching. The benefits of pre-fetching in this arena made it natural to apply this concept to the client/server model of the Internet. As a result, pre-fetching has garnered a great deal of recent attention as a possible solution for reducing latency and bandwidth consumption in World Wide Web queries.

As applied to web traffic, pre-fetching typically falls into two categories: pre-fetching content into a cache before the content is requested, and preparing connections to web servers before they are accessed. Examples of the first approach include The Web Collector [8] (a research prototype) and PeakJet2000 [6] (a commercial product), both of which preload into a local cache content associated with hypertext links of previously downloaded pages. Duchamp [2] and Bestavros [7] separately propose keeping statistics on the "relatedness" of web pages and their embedded links, and using these statistics to make decisions on pre-fetching. The second approach does not download content but instead prepares the connection that content downloads require. Cohen and Kaplan [1] consider methods for opening up HTTP connections to content servers in anticipation of use. The goal in this class of pre-fetching is to remove the latency associated with the overhead of establishing a connection.

3. Pervasive Content Delivery Systems

Most content delivery systems are designed to optimize overall system performance. They typically optimize system wide performance measures such as average bandwidth and average latency. In contrast, the design requirements for pervasive content delivery systems should be user-centric. Indeed, the main design goal of this paper is the reduction of the download latency experienced by each individual user to provide fast pervasive access to customized content. The term pervasive should again be highlighted to express our intent to deliver the right content at the right place and at the right time. Consequently, such systems must make use of two sources of information. The first source provides information from application providers, describing application semantics and the interdependency between the application components. The second source provides aggregate information on the end user. This is shown in Figure 1.

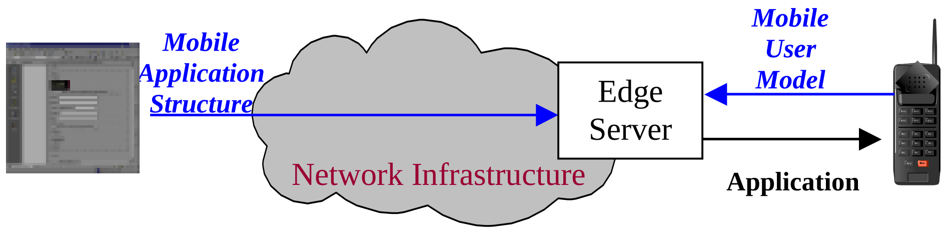

Figure 1: A Pervasive Content Delivery System. The edge server uses the application structure and a user model to intelligently predict usage.

The Application Model

In [3], it is argued that the heterogeneity of web documents precludes the use of a single caching or replication strategy for the management of all documents. Instead, the authors propose to attach a policy to each document in order to assist and optimize their delivery. We generalize this approach and propose the use of an application model that describes the complex relationship between logically related content objects.

The application model consists of an interface component graph with nodes connected to an arbitrary number of web services. The interface component graph is a directed graph representing the interaction and dependency between all the different modules of the application. For example, these modules can be HTML documents or Java Server Pages (JSP), connected to each other by the links that allow user to navigate between them. The web service could be a set of Enterprise Java Beans that dynamically provide content to both the HTML or JSP nodes of the interface component graph. A sample application model is shown in Figure 2.

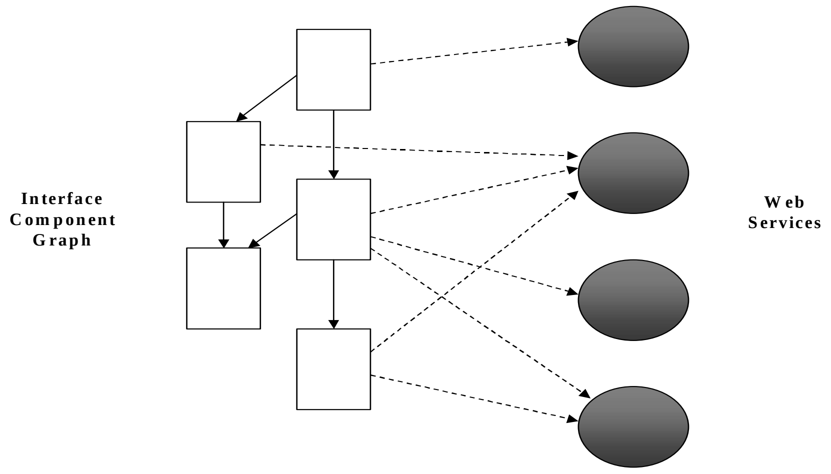

Figure 2: Application Model.

The User Model
The user model is a description of general properties of the end user. As shown in Figure 3, it has two main components:

- *Computational Capability:* Computation capability is the information about a user that is exploited in most existing content delivery systems. It includes standardized descriptions of the computational capabilities of all the devices present in the user's environment [5]. The description includes attributes such as screen size, color depth, storage capacity, CPU speed, etc.
- *User Context:* User context includes all the user characteristics related to his or her environment. We identify two subsets of context:
  - *User Preferences:* User preferences are a representation of the content access habits of the user. Preferences can be viewed as either deterministic or probabilistic. Deterministic preferences are obtained directly from the user via subscriptions (e.g., the user subscribes to a Dow Jones report), whereas probabilistic preferences are inferred by the system from the user's past request patterns.
  - *User State:* User state is composed of three attributes: the physical location of the user; the state of the applications that she is currently accessing; and which devices are active in her environment. These attributes define a virtual computer [4] composed of the active devices at the user's location.

4. Experimental Test-bed
We are developing a system to test our hypotheses about the extent to which pre-fetching content can reduce the latency of pervasive applications. Our initial experiments pre-fetch content based only on deterministic user models but not on application models. The main hypothesis that we are testing is that with a small amount of prediction on usage patterns of end users, preparing the edge of the network can significantly reduce the overall end-to-end delays observed by the users. Accordingly, the performance metric we employ is the average latency experienced by a user

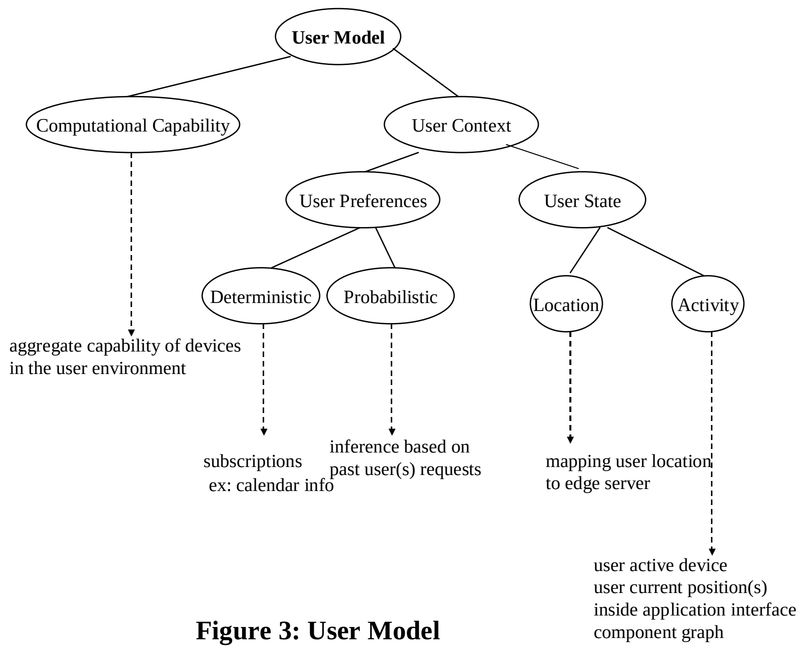
User Model
Computational Capability
User Context
User Preferences
User State
Deterministic
Probabilistic
Location
Activity
aggregate capability of devices in the user environment
subscriptions ex: calendar info
inference based on past user(s) requests
mapping user location to edge server
user active device
user current position(s) inside application interface component graph

Figure 3: User Model

when accessing a particular piece of content. A high level description of our test-bed is presented in Figure 4.

The main components of the test-bed are:

- *The Workload Generator:* This module generates two distinct datasets.
    - *User Subscriptions.* This is a set of deterministic user subscriptions of content, which is used by the pre-fetching module (below). This dataset consists of a four-dimensional matrix for each user: content requested, device, location and time of request. Ideally, this generation step should be the result of a random process with a distribution corresponding to the real distribution of pervasive users. However, this distribution is not well understood because of the lack of data on pervasive users. Thus, we focus on testing the correlation between the user model and the benefit of pre-fetching. We expect that this information will allow us to partition users into classes based on the benefit they receive from pre-fetching, leading to better models for pervasive application providers.
    - *User Requests.* This is a set of actual user requests for content, which is used by the client module (below). This dataset depends on the user's subscriptions. However, since content requests sometimes do not match previous predictions made by the pre-fetching module, (e.g., users occasionally need unanticipated content), this module randomly introduces variations to the user requests. The level of random variation is controlled by a parameter called the *access probability*. With this parameter, we are able to study the relationship between the accuracy of the user model and the latency penalty induced by incorrect pre-fetching operations.

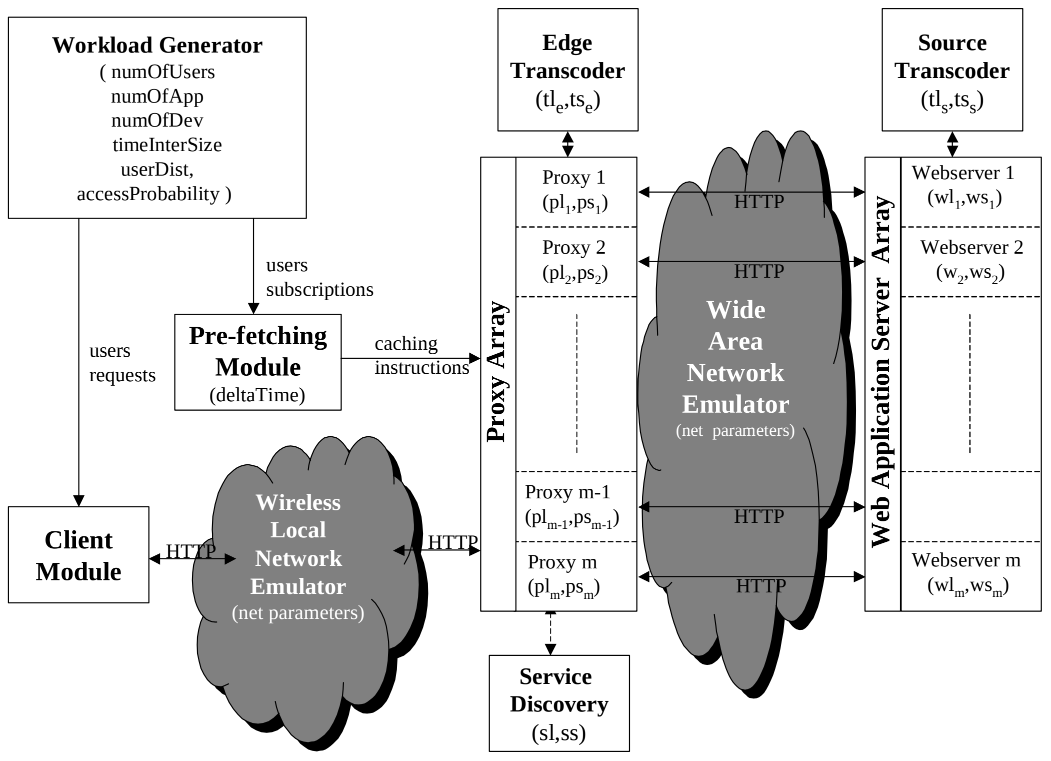

Figure 4: The main components of the test-bed. A description of the parameters shown in this figure is shown in Table 1.

- *The Client Module:* The client module emulates the behavior of end-users, by making requests from proxies on behalf of users. The behavior of users is determined by the User Requests dataset generated by the workload generator.
- *The Pre-fetching Module:* This module reads in the User Subscriptions dataset generated by the Workload Generator and instructs proxies to pre-fetch content in anticipation of client requests. Eventually, this module will incorporate various pre-fetching policies for probabilistic user models as well.
- *The Transcoders:* These modules adapt content to the capabilities of various client devices. The transcoding function can be invoked either at the source of the content or at the edge of the network.
- *The Service Discovery Module:* This module handles queries for services and returns service handles using an underlying repository of service handles. As described in Section 3, the application model uses of back end services to populate interface components with dynamic data. We have not integrated such capabilities into the test-bed yet and plan to address this point after studying the variability of access to static web content.
- *The Proxy Array:* Proxies are components that run at the edge of the network. Their functions include caching content, application offloading, binding applications to services, and invoking external modules such as for transcoding and service discovery mentioned above.
- *The Network Emulators:* The role of these modules is to emulate the performance dynamics in IP networks such as the Internet, and wireless networks such as IEEE 802.11 and Bluetooth. The emulation of dynamic IP networks is achieved by using the NIST Net

network emulator [9]. NIST Net is implemented as a kernel module extension to the Linux operating system with both an X Window System-based user interface application and a direct command line control. We have deployed NIST Net on each proxy module and each web server. The emulation of wireless network is also performed in a similar manner.

- *The Web Application Server Array:* These are standard HTTP servers that deliver applications as well as execute back end services (e.g., EJBs).

Workload Generator	numOfApp	number active of concurrent applications per user
	numOfDev	number of devices per user
	numOfUsers	total number of users
	timeInterSize	granularity of subscription length in time (tis = time increment size)
	userDist	user distribution
	accessProbability	Probability of accessing subscriptions
Pre-fetching Module	deltaTime	Difference between pre-fetching time and predicted acess time
Transcoders	tl_e,tl_s	proxy load respectively at the edge and at the source
	ts_e,ts_s	proxy speed respectively at the edge and at the source
Proxy	pl_m	proxy load
	ps_m	proxy speed
Service Discovery	sl	load on service discovery module
	ss	speed of service discovery module
Network Emulators	network parameters	packet loss rate bandwidth packet delay
Web Application Servers	wl_m	server load
	ws_m	server speed

Table 1: Parameters used to control the experimental test-bed

This test-bed is controlled by a set of parameters shown in Figure 4 and described in Table 1. The purpose of these parameters is to test our assumptions on the three factors affecting the latency within pervasive environments:

- Device heterogeneity. We test the effect of device heterogeneity by varying the physical location as well as the parameters associated with the transcoding proxies. Transcoding at the source versus the edge implies different loads on the web servers versus the edge servers, as well as different loads on the network.
- Network infrastructure. The effect of the network infrastructure is tested with the network emulators parameters such as bandwidth, packet loss rate and other network congestion parameters.
- User context. The last point tested is the variability of user context. As described in Section 3, the user context is a rich set of information about the user. The users location is modeled in this test-bed by their association to proxies in the proxy array and their movement among

them. The workload generator defines the location and trajectory of users. However, we are not yet studying the effect of discovering and binding services associated with the context of the user. We also do not track yet the activity portion of the user state (see Figure 3).

In brief, the current version of the test-bed contains all the mechanisms required to study the effect of variability on the delivery of static web content to pervasive users. As we continue experimenting with our test-bed we will extend it to incorporate the remaining characteristics of pervasive content delivery systems; namely the ability to predict usage patterns based on past behavior and the use of application models to make pre-fetching and offloading decisions at the edge of the network.

5. Concluding Remarks

We strongly believe that reducing access latency will be key to the deployment of pervasive applications. Pre-fetching content to the edge of the network close to the end user is the natural solution to this problem. In order to pre-fetch successfully we need:

- Knowledge of application structure.
- User models describing computational capability, user preferences and user state.
- Deployment of infrastructures at the edge of the network that intelligently pre-fetches relevant content for users.

We envision that technologies to support these issues will be at the heart of a middleware system for mobile computing. Our goal is to design such technologies.

Acknowledgements

The authors would like to thank Dr. Eric Nahum for the pointers he provided on how to develop the test-bed.

References

[1] Cohen, E. and Kaplan, H., "*Pre-fetching the Means for Document Transfer: A New Approach for Reducing Web Latency*", Proceedings of the IEEE Infocom 2000.

[2] Duchamp, Dan, "*Pre-fetching Hyperlinks*" 2nd USENIX Symposium on Internet Technologies & Systems, Boulder, CO, October 1999.

[3] G. Pierre, I. Kuz, M. van Steen, A. S. Tanenbaum, "*Differentiated Strategies for Replicating Web Documents*", Computer Communications 24 (2001) 232-240

[4] R. Han, V. Perret, M. Naghshineh, "*WebSplitter: Orchestrating Multiple Devices for Collaborative Web Browsing*", ACM Conference on Computer Supported Cooperative Work (CSCW), December 2, 2000

[5] "*Composite Capabilities/Preference Profiles: Requirements and Architecture*", W3C Working Draft 21 July 2000, http://www.w3.org/TR/2000/WD-CCPP-ra-20000721/

[6] PeakJet2000 Software, http://www.peak.com/peakjet2long.html

[7] A. Bestavros, "*Using Speculation to Reduce Server Load and Service Time on the WWW*", Proceedings of the 4th ACM Intl. Conf. on Information and Knowledge Mgmt. ACM, November 1995. http://www.cs.bu.edu/~best/res/papers/cikm95.ps

[8] The Web Collector, https://www.inkey.com/save30/

[9] NIST Net Home Page, http://snad.ncsl.nist.gov/itg/nistnet/