Rochester Institute of Technology

RIT Digital Institutional Repository

Theses

3-1-2013

An Efficient interaction framework for mobile web services

Abdullah Almuaibid

Follow this and additional works at: https://repository.rit.edu/theses

Recommended Citation

Almuaibid, Abdullah, "An Efficient interaction framework for mobile web services" (2013). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by the RIT Libraries. For more information, please contact repository@rit.edu.

An Efficient Interaction Framework for Mobile Web Services

by

Abdullah Abdullatif Almuaibid

A Thesis Submitted
in
Partial Fulfillment of the
Requirements for the Degree of
Master of Science
in
Computer Science

Supervised by

Xumin Liu, Ph.D.

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

March 2013

The thesis “An Efficient Interaction Framework for Mobile Web Services” by Abdullah Abdullatif Almuaibid has been examined and approved by the following Examination Committee:

Xumin Liu, Ph.D.
Assistant Professor
Thesis Committee Chair

Rajendra K Raj, Ph.D.
Professor

Stanislaw P Radziszowski, Ph.D.
Professor

Dedication

To all who have the thirst to improve this work..

Acknowledgments

This journey that I took in pursuing my studies was a blessing, and I am grateful for having the opportunity to do so. I am thankful for all the encouragement and help I received from my family, professors, and friends.

Abstract

An Efficient Interaction Framework for Mobile Web Services

Abdullah Abdullatif Almuaibid

Supervising Professor: Xumin Liu, Ph.D.

Having the latest technological devices is becoming more of a necessity these days rather than a luxury as it was only a decade ago. Among these advancements, the smart phone has grown in popularity significantly. This is due to a variety of reasons but it would have to be said that the access to mobile internet service it provides is a strong motivator for acquiring one. As might be expected the increased number of smart phone users requesting information from the internet has resulted in times when the *average access time* (the time a user must wait to download his/her requested service) is quite high. In research the use of the wireless carrier's broadcast channels to transmit mobile web services has proven to be an economical way to handle the increased volume of users. However, having not been designed for this purpose, the usual configuration of the broadcast channels fails to take into account the characteristics of today's devices and those of mobile web services. Therefore, this thesis will outline the formulation of an interaction framework that provides an innovative channel design and allocation approach for delivering mobile web services. This architecture will incorporate current usage data (*i.e.,* popularity over specified time period) and the exact dimensions of each mobile web service to more efficiently allocate them, thus resulting in lower *average access time*.

Contents

List of Figures

Chapter 1

Introduction

A smart mobile device has become something many people "can't live without" [5]. This is not surprising as it affords them the ability to be connected to the entire world and in general is like having an omniscient personal assistant thanks to its capability of accessing the internet. The demand for web access by mobile device users has expanded exponentially in recent years. With the continued investments being made by wireless carriers in advanced telecommunication technologies such as WiFi, WiMaX, UWB, 3G/UMTS, and 4G LTE, there has been an increase in the availability of bandwidth for use by these mobile devices. Additionally, broadcast is considered an effective approach for delivering information to mobile users. It facilitates the distribution of mobile web services (MWSs) among mobile devices and the wireless network. Due to its scalability, the number of users does not affect the response time for retrieving the information. Also advantageous is the fact that broadcast does not consume significant amounts of power on the client side [7, 9, 11, 12].

"A web service is a software system designed to support interoperable machine-to-machine interaction over a network" [6]. This type of software (which is usually carried by HTTP) has several advantages such as interoperability, usability, reusability and deployability. Web services have significantly improved numerous aspects of using computers to access and then process data in other machines. To be more specific, web services rely on three core components: SOAP, WSDL, and UDDI. All of these are Extensible Markup Language-based (XML). The Simple Object Access Protocol (SOAP) component consists of two parts: the Header, which has the processing information of the web services and the Body which contains the information users want to process. The Web Services Description

Language (WSDL) is the second core component; it allows the service requester's device to know how to deal with (invoke) services from the service provider without knowing the exact specifications of each other's programs. Finally, the Universal Description, Discovery and Integration (UDDI) is the third component; its function is to carry the WSDL for the web service provider in order to share and advertise its services to all the service requesters who may be interested. Figure 1.1 represents the life cycle of a web service (Web service reference model).

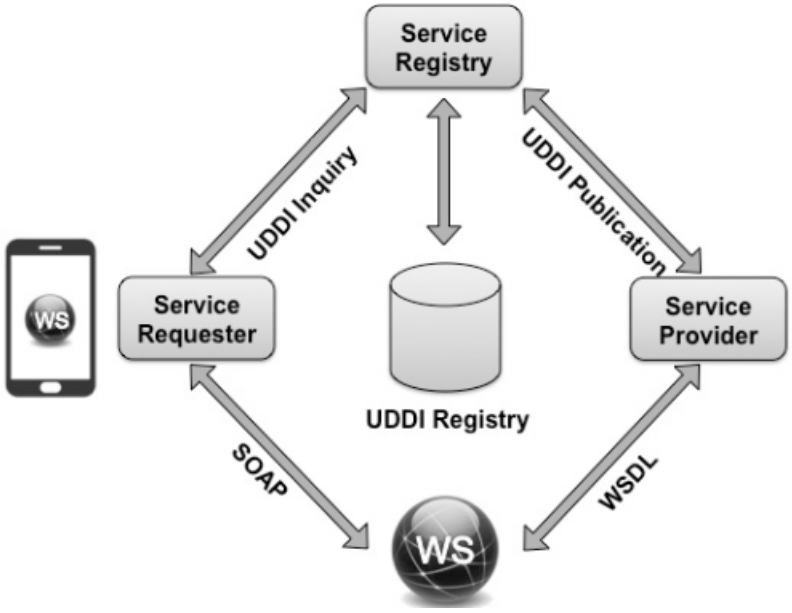


Figure 1.1: Web service reference model.

In general a broadcast mobile web services' infrastructure [10] consists of three parts: *Web Service Providers, Wireless Carriers and Mobile Users*. Customarily, mobile web services are designed by web service providers and provided to wireless carriers (*e.g.,* AT&T, T-Mobile and Sprint). The communication among these parties is usually passed through wired media. In this case, a web service provider can come in many forms, such as a person, company or an organization. By developing mobile web services and publishing them to wireless carriers (usually a telecommunication company that owns cell phone networks)

the web service providers are able to disseminate information about their web service offerings to mobile device users. In turn, their potential customers can access those same cell phone networks and request any mobile web service (ideally) through the wireless carrier's cell phone network's broadcast channels. Figure 1.2 represents the life cycle of a mobile web service.

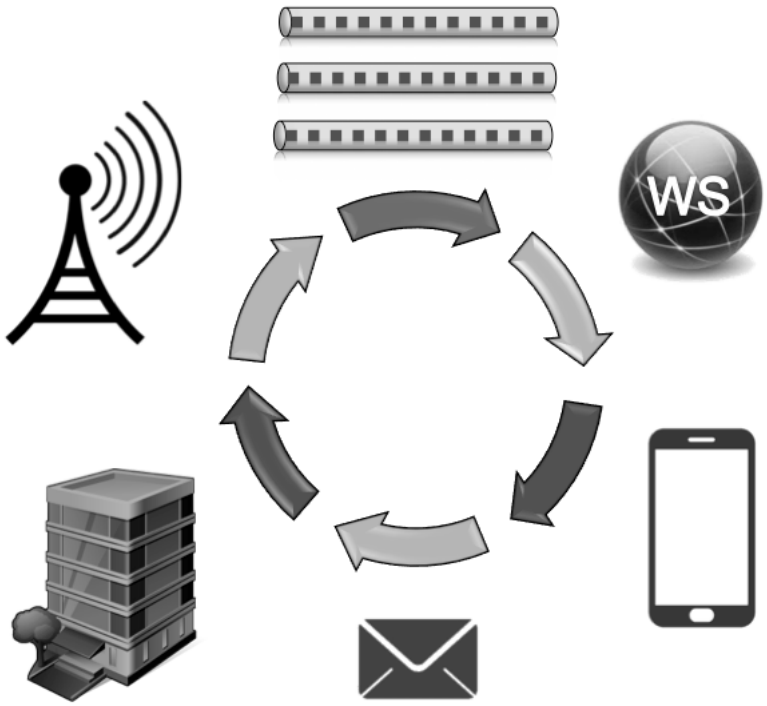


Figure 1.2: Life cycle of a mobile web service.

In this scenario, each of the participants (*e.g.,* web service providers, wireless carriers and mobile users) has its own tasks or operations to execute. For instance, web service providers have the responsibility to *Publish Services*; that is to supply detailed information about their mobile web services and how to request them. They also *Provide Live Data Feeds* which means to transmit up-to-date data feeds to the wireless carriers. And finally, web service providers pass along *Service Updates* which inform the wireless carriers of changes to any of their mobile web services and also include updated registry information.

On the other hand, wireless carriers have the ability to remove any mobile web services from their network's broadcast channels by notifying the corresponding web service

provider. Wireless carriers are in charge of *Defining and Providing Service Provider Interfaces*. Once established, the web service providers can publish and manage their services through this interface. Additionally, the wireless carriers are responsible for *Providing the Mobile Web Services' Registry*. It is similar to a database in that it maintains all available mobile web services and each of their corresponding registry data. Finally, wireless carriers are tasked with *Organizing and Broadcasting Services and Data*. Their goal is to arrange the broadcast channels so that all necessary data pertaining to their mobile users' requested services is readily available.

The final element in the mobile web services' infrastructure is the mobile users. In order for them to be able to use mobile web services they must have the capability of being *Wireless Receivers*. That is to say, at least one wireless receiver must be present in the mobile device for the purpose of interacting with the wireless carrier's network of broadcast channels. Finally, the mobile devices need to have the functionality of *Service Invokers*. By this we mean, they must be able to interpret the registry, request their desired service, access the service and return some output if needed.

The objective of this thesis is to design and subsequently evaluate an efficient interaction framework (which will be comprised of a broadcast channel design and an allocation approach) for delivering mobile web services in a wireless environment. As the basis for this scenario, it is assumed that the wireless carrier will use its cell phone broadcast network to transmit mobile web services to its customers [7, 9, 8, 12]. This proposed architecture allows for the broadcast of mobile web services via a set of pre-defined broadcast channels. In addition to this prescribed channel configuration, the concept of "skewing the data" will be uniquely applied to the allocation of broadcast channels based on data demand [7, 9]. The desired outcome of this thesis is to produce efficiencies on the mobile web service provider side and on the mobile user side (*e.g.,* faster average access time, lower power consumption and processor utilization, *etc.*). Specifically, for the mobile web service consumer, we seek to reduce usage of the mobile device's limited resources by allocating mobile web services efficiently into broadcast channels and thus allowing them to be accessed more quickly

(and simultaneously) by a multitude of mobile device owners.

Chapter 2

Related Work

With a firm understanding of the mechanics of web services, further (more specific) research on mobile web services provided us with novel approaches and valuable "food for thought". One such example is the concept of accessing web services via cell phone broadcast networks [7, 9]. Another paper describes different broadcast channel structures which were invented to share any data through a cell phone broadcast network [9]. Yet another research introduces the concept of *skewing the data* to accommodate the fluctuations in the number of requests for different types of data [7, 9, 11].

Not surprisingly, a substantial number of researchers have included the use of broadcast channels in their mobile web services solutions due to the advantages of scalability, energy efficiency and unlimited user capacity that they bring to the table [9, 7, 11, 1] The challenge becomes the allocation of the data among the broadcast channels so that the *average access time* (AAT) is consistent (regardless of the "popularity" of the data requested). One approach is to allocate bandwidth to the transmission of less frequently requested data while leaving broadcast to handle the more popular data as was suggested in [2] and [12]. A different concept is to send out the most popular information requests at increased intervals on the broadcast channel, thereby skewing the data allocation [11].

In essence, Yee *et al.* 's [11] research is based on the theory that by using multiple broadcast channels (instead of the usual single channel design) response time between mobile devices and web service providers will be improved and the added benefits of improved "fault tolerance, configurability and scalability" could be realized. The team's unique approach called for partitioning the data (and scheduling it) among several broadcast channels

with their simplified GREEDY approximation algorithm. In their experiments, they compared the performance of their algorithm (GREEDY) to FLAT, Bin-Packing (BP), VF^K and Dynamic Programming (DP) on data items of the same size. Their goal was optimization balanced with practicality thus taking into consideration the limitations of the on-line and mobile environments. As they had hoped, the GREEDY algorithm produced near-optimal performance within its less complex (when compared to previously used algorithms) operational mode [11].

Tackling more complicated wireless transactions, Yang *et al.* [9] sought to apply efficiencies to accessing composite mobile services (a service which depends on other services to work) which are defined by Business Process Execution Language (BPEL) graphs. They discovered that the *closest-first-access* algorithm had advantages over the *depth-first-access* and *breadth-first-access* algorithms no matter the type of composite service (*i.e.,* sequential, parallel or hybrid). Due to the intricacies of these operations, the team sought to breakdown the access method into three processes: service request flow, receiver flow, and service execution flow. Their proposed broadcast channel configuration was made up of seven channels with each having its own designated purpose. More specifically, they were each assigned a certain type of content, such as M-services, wireless data, service descriptions, composite services, or service registry information.

Additional efficiencies for the above configuration were sought by the team through "selective tuning" which would allow the mobile devices to rest (doze) and only activate when its required content became available. This was accomplished by developing their "index channel method" which consisted of a *M-service index channel* and a *data index channel*. Both channels provided service or data location information ("arrival time") in index format to the mobile device which allowed it to doze while waiting and thus reduced its tuning time (and energy consumption).

Another of the earlier broadcast-based solutions was one proposed by Zheng *et al.* [13] who were the first to attempt to accommodate K Nearest Neighbor (K-NN) requests by mobile device users. As part of their process, they chose to utilize "interleaving an

auxiliary index with data" as an energy-saving mechanism (*i.e.,* reducing tuning time) in an effort to achieve results similar to those in [9]. Other challenges which they sought to overcome included the random access storage-based nature of K-NN search indexes and the need to "packetize" information for transmission to the client.

In the research above (and historically speaking) those designing and delivering mobile web services have had to contend with the limitations (*i.e.,* battery life, bandwidth and storage) which are inherent in most mobile devices. Contrary to this thinking, Chou and Li [4] designed a mobile distributed computing platform (based on Android technology) which facilitates distributed services computing on mobile devices utilizing real-time and over IP communication capabilities. Their Web Service Initiation Protocol (WIP) enables mobile devices to act like/appear as web service providers. Along the same lines, Aijaz *et al.* [3] set out to prove that mobile devices could host asynchronous mobile web services (Mob-WS). Their scenario had web service requestors and web service providers both being mobile devices. They established Peer-to-Peer communication between the parties by leveraging Bluetooth technology®.

All of the research mentioned above and what has been published up until the date of this thesis have not considered utilizing skewed hybrid broadcast channels (disseminating variably-sized mobile web services) to reduce *average access time*. This thesis is devoted to the design and execution of this conceptual model which includes its innovative channel design and allocation approach.

Chapter 3

Hypothesis

As discussed previously, past researchers have chosen to utilize cell phone wireless carriers' broadcast networks to deliver mobile web services to their subscribers. This proved to be an advantageous and efficient approach in many scenarios [9, 7, 12, 8]. One of the main benefits for pursuing this strategy is that broadcast is not affected by the number of mobile users who receive data via this portal. However, the mobile users do have to wait for their specific mobile web service to arrive in the broadcast channel in order to be able to access it. Other researchers came up with a design which skews the data distribution in the cell phone networks' broadcast channels [11]. The theory underlying our paper is that a new interaction framework could be achieved by having a unique architecture for the broadcast channels and dynamically skewing the mobile web services into their specified channel(s) (while still taking into consideration their popularity). As a result, the desired configuration is a more efficient channel design and allocation system for delivering mobile web services to the many of mobile users who desire them.

This proposed architecture (see Figure 3.1) consists of two main components. One is the *mobile application*, which among other things, facilitates the user's ability to explore the available mobile web services. Additionally, it allows the mobile user to communicate automatically with the wireless carrier's broadcast server. This line of communication is necessary because, as we know, the broadcast channels themselves can only communicate in one direction. Therefore once the mobile user downloads his/her desired mobile web service, the mobile application will send a *miniscule acknowledgement* directly to the broadcast server so that the user's choice will be recorded. (It's important to note that this

acknowledgement is transmitted as anonymous data as knowing which mobile user is located where and what mobile web server he/she is using/requesting could be considered an invasion of privacy.) These acknowledgements are being tabulated for the purpose of determining the popularity and frequency of each web site's usage. The data will be used in the calculations which determine the broadcast channel scheduling.

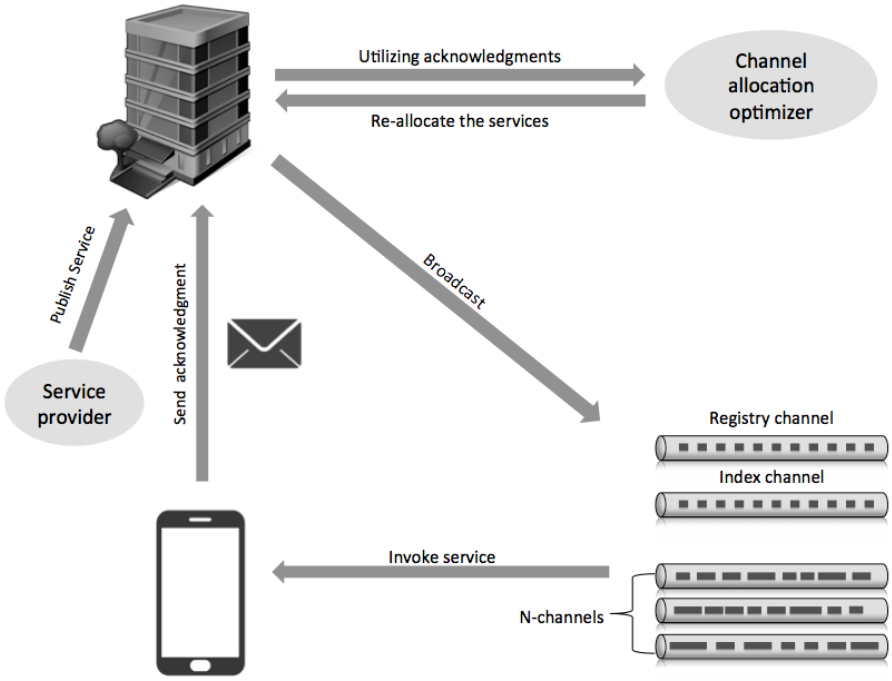


Figure 3.1: Mobile web service reference model

The other main architectural component is the *broadcast channel configuration*. It consists of three core channels: a Registry channel, an Index channel and a Hybrid channel.

The following are descriptions of each type of main channel:

1. *Registry Channel*: This channel (Figure 3.2) functions like a menu in that it is a listing of all the mobile web services which are available in the geographic area in which the mobile device is located. When first entering a region, the mobile device's mobile web services application will download and store the contents of the registry file which will be used as a reference when looking for mobile web services in this vicinity. Additionally the registry channel continuously disseminates "registry records"

(*e.g.,* from the wireless carrier's UDDI), each which contains basic, individual mobile web service information: name, category type (*e.g.,* weather, sports, dining, *etc.*), and a short description. Additionally, at the beginning of each registry record is found the mobile web service's "service key" which is a unique number assigned to it for identification purposes. (This service key will be utilized by the mobile device in subsequent searches in order to eventually obtain its selected mobile web service.) The comparatively small size of these registry records makes storing them on a mobile device reasonable and necessitates only one broadcast channel being designated for their transmission.

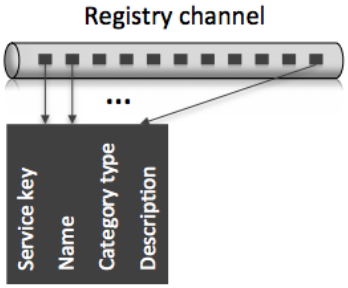


Figure 3.2: Registry channel

2. *Index channel*: This channel (Figure 3.3) provides a listing which informs the mobile device user where its desired mobile web service will be broadcasted (*i.e.,* in which of the many hybrid broadcast channels). Streaming continuously through this broadcast channel are the service key and the corresponding hybrid channel location for each of the available mobile web services. Attached with the service key (from the registry channel) for its chosen mobile web service, the mobile device "consults" the index channel for the designated number of the hybrid channel where this mobile web service's components will be broadcasted. From the diagram below, you can see that this channel too contains very small amounts of data and therefore only one index channel is required in this architecture. Efficiencies are realized from this arrangement as a single index channel requires less tuning time and average access

time than multiple index channel configurations (*e.g.,* service index and data set index in [9].

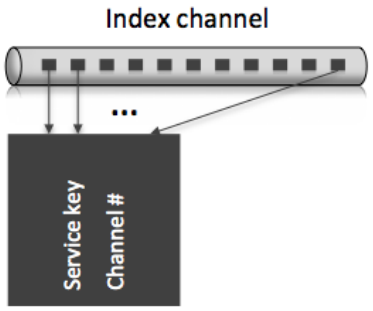


Figure 3.3: Index channel

3. *Hybrid channel*: The "hybrid" channel (Figure 3.4) is so named because it transmits each mobile web service with its associated service key, WSDL and corresponding data. Similar to the index channel, each record begins with the service key identifying number. The "combined" construction of the hybrid channel also reduces the tuning time and average access time from what is needed when separate channels are used for the broadcast of the various mobile web services' components (as in [9, 7]). To elaborate, the mobile device tunes to the hybrid channel number which was listed in the index channel directly following the service key for its selected mobile web service. Since the hybrid channel is broadcasting in the pattern of: service key(a), mobile web service(a), WSDL(a), data set(a), service key(b) ... the mobile device can "doze" during most of the transmission and only activate when a new service key arrives in the channel. (As you can see, this reduces the tuning time needed to find the mobile web service.) If it does not correspond to the service key number that the device is looking for, it will doze again until the next service key appears in the hybrid channel. When the matching service key number is broadcasted, the mobile device will download the related mobile web service, WSDL and corresponding data and be able to invoke the service.

   The contents of the hybrid channels are dynamically updated (skewed) based on two

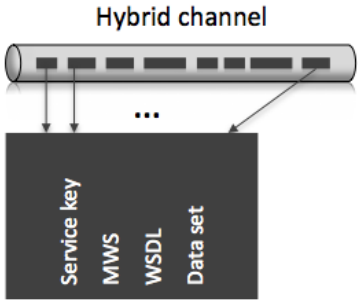


Figure 3.4: Hybrid channel

factors/thresholds:

(a) *Period of time" (t)* e.g., *3 hours and*;

(b) *Pre-defined number of requests/user acknowledgments, (r)* e.g., *1000*

These two attributes determine the most highly demanded mobile services and assign them to the hybrid broadcast channels based on that, with the result being that the most popular will be in less crowded channels. It's important to note that the mobile web service re-allocation process (among the hybrid channels) will also result in the index channel records being changed as some of the mobile web services maybe being broadcast in different hybrid channel locations. When the user has successfully reached his/her desired mobile web service and downloaded it, the mobile application will send an anonymous acknowledgement to the broadcast server, which will add it to the total number of acknowledgements already received for that particular mobile web service.

Generally speaking, the example that follows will illustrate how this solution will function. Hypothetically, we have six randomly-sized mobile web services ($ws_1, ws_2, ... , ws_6$); each of them will have a counter assigned to it (associated with the "maximum number of requests" threshold). If a user downloads one of the mobile web services, the mobile application will send an anonymous acknowledgment to the broadcast server and its counter will increase by one (*i.e.,* total number of requests for that mobile web service).

In one scenario, the "maximum number of requests" does not meet the threshold but the

other threshold (which is “period of time”) has been met. In this case, the algorithm will reassign the mobile services to broadcast channels due to the fact that the “period of time” was exceeded. In the second scenario, the threshold for “maximum number of requests” has been met before the “period of time” expired and therefore the algorithm will reorganize the mobile services to different channels based on the quantity of requests (*i.e.,* number of user acknowledgments received). The two thresholds ensure that the allocation of mobile web services is based on real-time(*i.e.,* up-to-date) users’ requests.

The proposed architecture, with its two main components, plays an important role in reducing the number of ticks (*i.e.,* the time measurement for each “bucket” where each mobile web service could occupy several buckets) the user needs to connect with the required broadcast channels [11]. Hence, reducing the average access time experienced by the user before the desired mobile web service is available for download.

Chapter 4

Different Allocation Approaches

During the course of this research, we have been testing different types of allocation methodologies to determine the "cheapest" (*i.e.,* least expensive to execute for all parties involved) allocation approach which still served our purposes (*i.e.,* shorter average access time & utilizing fewer mobile device and wireless carrier resources). The three different allocation approaches that we used were:

1. Formulating all possible combinations and sequences (*i.e.,* Brute Force),
2. Sorting the mobile web services and allocating them heuristically (*i.e.,* Greedy) and finally
3. Using Dynamic Programming to allocate them

It proved to be that it was our second approach (*i.e.,* Greedy) which had better results than the first approach and it is significantly less expensive to use. In contrast, an example involving allocating mobile web services using the "Brute Force method" with only three hybrid channels and five mobile web services a total of 25 MWS allocation permutations were produced. Furthermore, by adding one additional MWS to the same configuration (*i.e.,* using three hybrid channels), the number of possible MWS allocation arrangements rises to 105. For each of these combinations a calculation must be made in order to determine which one produces the lowest AAT. The bottom line is that each additional MWS added causes the number of calculations to go up substantially and along with it, the costs (and time) associated with this method also increase. Similarly, the Dynamic Programming

approach proved to consume much more memory space and time than the Greedy-based process. Chapter 6 provides examples and details of these outcomes.

Chapter 5

Problem Statement

5.1 Single channel average access time

In a single channel, one or more mobile web services could be there. Therefor, we noted the MWS as C_m where m is the number of mobile web services in the channel and it is of a range $1 \leqslant m \leqslant C_m$. L_i is the length of a mobile web service and its associated data (how many buckets it occupied on the channel). And W_i is the weight for a certain mobile web service (as determined by the number of *acknowledgments* generated by mobile users who accessed that particular service %75 and the size of it %25). The total length of the channel is denoted as T. To determine the *average access time* for a mobile web service in a single hybrid channel configuration, we used the equation below (5.1).

$$\sum_{i=1}^{C_m} \left(\frac{2L_i + T - 1}{2} \times W_i \right) \tag{5.1}$$

Taking into account that mobile web services are cyclically broadcasted into the channel, this equation (5.1) calculates the average access time by adding together the "best case scenario" of accessing a mobile web service and the "worst case scenario" of accessing the same mobile web service and then calculating the average.

5.2 Multichannel average access time

In this scenario, there are K hybrid channels, each noted as Ch_i with a range of $1 \leqslant i \leqslant K$. C_m is the number of mobile web services in a single channel (*i.e.,* Ch_1), each of the

channels noted as C_m where m is of a range $1 \leqslant m \leqslant C_m$. L_i is the length of a mobile web service and its associated data while W_i is the weight for a certain mobile web service. In order to determine the *average access time* of a mobile web service in a multiple hybrid broadcast channel arrangement, we used this equation (5.2).

$$\sum_{i=1}^{K} \left(\left(\frac{\sum_{j=1}^{C_m} L_j}{2} - 1 + L_j \right) W_i \right) \tag{5.2}$$

5.3 Methodology

In general our two-component architecture (*i.e.,* mobile application and broadcast channel configuration) plays an important role in reducing the *average access time* experienced by the user (*i.e.,* before his/her desired mobile services can be accessed). Delving into the details, you will find a broadcast channel allocation process which is based on real-time requests (popularity) of mobile web services which are accounted for by the accumulation of user 'acknowledgements'. To this continuously updating scheduling procedure, the addition of a time threshold has been incorporated. This feature permits the distribution of mobile web services to be based on/limited to a specific timeframe. When the specified time has elapsed:

1. the 'counters' are re-set,
2. the current popularity can then become "visible" again and
3. the allocation is updated.

The 'period of time' threshold prevents the possibility that a mobile web service will be given priority based on out-of-date accessing data.

As a starting point, we begin with calculating the *average access time* for requesting a mobile web service from a single broadcast channel by calculating the average (*i.e.,* the best case and the worst case scenario added together and taking the average) as can be seen in equation (5.1).

In contrast to previous research, our mobile web services and their corresponding data sets are variable in size. Therefore each is transmitted in a series of uniform-sized “buckets” and allocated to one of several hybrid mobile broadcast channels. Thus we needed to adjust the equation above (5.1) to take into account the non-uniform size of our test data. The result is the updated equation (5.2).

Chapter 6

Framework Design

In order to examine and test the results of our proposed framework, we use *average access time* for measuring the retrieval time of specific mobile web services from the broadcast network channels. We proposed the use of a skewed broadcast channel architecture, however in contrast to [11], mobile web services data are utilized instead of their generic, uniform-sized data. Each of our mobile web services has associated data and a request counter (to tabulate *users' acknowledgments*) in order to more closely mimic a 'real world' scenario. The result is that each mobile web service is a different size and thus requires a variable number of "buckets" in order to transport it inside the broadcast channel.

Three allocation methods were used during our testing period: Brute Force, Greedy and Dynamic Programming. Therefore, the results included are the recording of *average access time* for retrieving mobile web services through skewed channels in these three allocation configurations. The purpose of calculating AAT for the various permutations of allocating the available MWSs is to determine the specific configuration which produces the least expensive way to calculate AAT (thus most cost-effective arrangement) arrangement. Based on these findings the MWSs are re-allocated among the available broadcast channels. Figure 6.1 shows the architecture of the skewed broadcast channels.

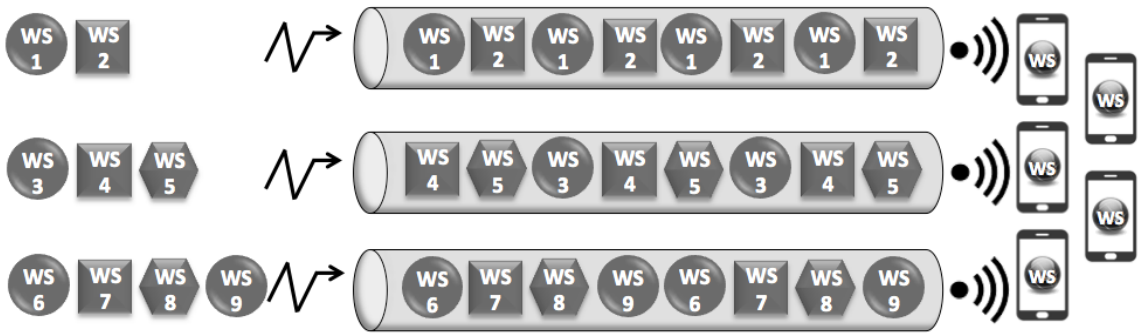


Figure 6.1: Architecture of the skewed broadcast channels

The Brute Force algorithm, Greedy algorithm, and Dynamic Programming algorithm allocation is described below:

6.1 Brute Force

The Brute Force algorithm-based allocation scheme proved to be the most expensive alternative among the three (*i.e.,* Greedy, Dynamic and Brute Force). In the first example below you will see that this allocation methodology is inconvenient and time consuming as it produces an inordinate number of allocation combinations even when dealing with a small number of mobile web services. Example 1 configuration:

- 3 hybrid broadcast channels
- 5 mobile web services (designated “a”, “b”, “c”, “d” & “e”)

Based on the assumption that none of the broadcast channels will be left empty (*i.e.,* a minimum of one mobile web service in each channel), only two basic distribution formulas are possible: $3/1/1$ and $1/2/2$. Allocation $3/1/1$ produces the following possible combinations (see Table 6.1)

Total: 10 combinations

Allocation$1/2/2$ produces the following possible combinations (see Table 6.2)

Total: 15 combinations

Table 6.1: All possible combinations for $3/1/1$

abc	d	e
abd	c	e
abe	c	d
acd	b	e
ace	b	d
ade	b	c
bcd	a	e
bce	a	d
bde	a	c
cde	a	b

Table 6.2: All possible combinations for $1/2/2$

a	bc	de
a	bd	ce
a	be	cd
b	ac	de
b	ad	ce
b	ae	cd
c	ab	de
c	ad	be
c	ae	bd
d	ab	ce
d	ac	be
d	ae	bc
e	ab	cd
e	ac	bd
e	ad	bc

As you can see this relatively small-sized example has produced a grand total of 25 mobile web services allocation permutations which, according to our process, will each have to be tested to determine which is the most efficient mobile web services distribution.

In the second example (see below) we increase the number of available mobile web services to six (*i.e.,* "f") to see what effect this may have on the number of possible allocation combinations. The number of available hybrid broadcast channels remains at three. The six mobile web services now generate three possible basic distribution formulas: $4/1/1$, $3/2/1$ and $2/2/2$.

A $4/1/1$ allocation is represented in the formula below:

$$\frac{\binom{6}{4} \times \binom{2}{1} \times \binom{1}{1}}{2} = 15 \tag{6.1}$$

Total: 15 combinations

A $3/2/1$ allocation is represented in the formula below:

$$\binom{6}{3} \times \binom{3}{2} \times \binom{1}{1} = 60 \tag{6.2}$$

Total: 60 combinations

And a $2/2/2$ allocation is represented by the formula below:

$$\frac{\binom{6}{2} \times \binom{4}{2} \times \binom{2}{2}}{3} = 30 \tag{6.3}$$

Total: 30 combinations

With this second example, we can see that changing just one of the factors slightly (*i.e.,* number of MWSs) results in an increase to 105 allocation permutations (vs. the 25 combinations in example number 1). This is conclusive proof that the Brute Force algorithm based allocation scheme is an inefficient methodology for our purposes.

6.2 Greedy

Our second testing scenario involved the use of a Greedy algorithm. As our starting point we began with the Greedy algorithm presented in [11]. Since our data was significantly different than theirs, we modified their formula to take that into account. Among the changes we made was the establishment of a weighting criteria which was utilized when all of the available mobile web services are first sorted. Rather than just rank them strictly by the number of user acknowledgements received ("popularity") we also took into account the amount of space that each mobile web services would take up in the broadcast channel ("length"). For our purposes, we weighted popularity ratings as 75% and size of mobile web services as 25% in our ordering process and the cost model. As you can see from the steps below, the Greedy algorithm determines the possible combinations of mobile web services and the *average access time* associated with retrieving each. After an analysis of the results is made, the re-allocation of the mobile web services into the available broadcast channels will be made based on the configuration which gives the lowest AAT. Follows is an example representation of the Greedy algorithm. Below the example is the pseudocode for the same algorithm.

After sorting six MWSs based on weight, the allocation or the configurations of Greedy for three channels will be as shown below:

MWS3 MWS5 MWS1 MWS4 MWS6 MWS2

Greedy will start from the first MWS *e.g.,*

MWS3 MWS5 MWS1 MWS4 MWS6 MWS2

and consider it as the best splitting point. This process will go through all the possible splitting points *e.g.,*

MWS3 MWS5	MWS1 MWS4 MWS6 MWS2
MWS3 MWS5 MWS1	MWS4 MWS6 MWS2
MWS3 MWS5 MWS1 MWS4	MWS6 MWS2
MWS3 MWS5 MWS1 MWS4 MWS6	MWS2

Based on the lowest AAT provided from the multi-channel equation, one of the configurations will be chosen.

If this configuration is chosen:

MWS3 MWS5 MWS1 MWS4 MWS6 MWS2

The second step is to do the same for both sides and decide which configuration will provide the lower AAT.

MWS3 MWS5

MWS1

MWS4 MWS6 MWS2

Verses

MWS3

MWS5

MWS1 MWS4 MWS6 MWS2

Because the target is three channels, passing these two configurations above to the multi AAT equation will determine which one is the best allocation.

Algorithm 1 Greedy

```
1: Parameters:
2: listOfMWS: an ArrayList of input objects
3: numberOfChannels: number of channels
4: numberOfCurrentChannels: initial as 0
5: Sorts all available mobile web services (MWS) (found in ArrayList listOfMWS) based
   on weight.
6: while numberOfCurrentChannels < numberOfChannels do
7:    for first index of the service in the channel to index of last service in the channel  do
8:       calculate and check which index is the best splitting point using SAAT
9:       Break the initial list up into two channels based on the provided splitting point
10:      The first split will yield two channels. Those channels will now become keys to
         the hash map, which will point to the new channels they will form
11:      calculate AAT for the current sub-channels to identify which has the lowest AAT
         using MAAT
12:   end for
13:   choose the splitting that yields the lowest AAT
14:   numberOfCurrentChannels += 1
15:   if numberOfChannels == numberOfCurrentChannels then
16:      Exit
17:   end if
18: end while
```

6.3 Dynamic Programming

For our third testing approach, we chose Dynamic Programming as they did in [11]. Similar to our procedure with the Greedy algorithm test, we began with the DP algorithm from Yee *et al.* [11]. With this as our base, we made adjustments to it due to the random (variably-sized) nature of our data and its composition (actual MWS data vs. generic data). The same test data that was run through the Greedy algorithm was then processed through the updated DP algorithm. The result was that DP's allocation configuration produced the optimal solution. This is not surprising if you consider the nature of Dynamic Programming. Below, you will find DP algorithm equation 6.4. Following that is an example of the steps that Dynamic Programming goes through in its search for the best outcome (in this case lowest AAT possible for all available MWSs).

After sorting six MWSs based on weight, the allocation or configurations of DP for three channels will be as follows:

MWS3	MWS5	MWS1MWS4MWS6MWS2
MWS3	MWS5MWS1	MWS4MWS6MWS2
MWS3	MWS5MWS1MWS4	MWS6MWS2
MWS3	MWS5MWS1MWS4MWS6	MWS2
MWS3MWS5	MWS1	MWS4MWS6MWS2
MWS3MWS5	MWS1MWS4	MWS6MWS2
MWS3MWS5	MWS1MWS4MWS6	MWS2
MWS3MWS5MWS1	MWS4	MWS6MWS2
MWS3MWS5MWS1	MWS4MWS6	MWS2
MWS3MWS5MWS1MWS4	MWS6	MWS2

Each allocation will be calculated once, if it is a single channel (*e.g.,* MWS3 or MWS3MWS5). This configuration will be plugged into the single channel equation. On the other hand, each line is considered one allocation for the three channels (*e.g.,* MWS3, MWS5, and MWS1MWS4MWS6MWS2) and will plugged into the multi-channel equation. A sum for each channel will be calculated, and the allocation will be chosen based on the lowest AAT.

$$dpOptSol_{i,K} = LAAT(AAT_{im} + dpOptSol_{m+1,K-1}) \tag{6.4}$$

K = number of the hybrid channels

$LAAT$ = lowest AAT

m = total number of MWSs-1

i = start index of MWSs in a channel ($1 \leq i \leq m$)

j = end index of MWSs in a channel ($i < j \leq m$)

In the Figure 1.2, we can see the mobile web services cycle as it is proposed here:

Starting on the left we have the wireless carrier (A) utilizing its broadcast network (B) to disseminate the mobile web services it has to offer. Similar to pipelines, there are multiple broadcast channels (C) in the network which transmit mobile web services at different intervals. Mobile devices (E) entering the broadcast networks geographic area are apprised of the web services which are available in this region.

By clicking on a choice, the mobile user will then download his/her desired web service (D) to the mobile device (E). In this solution, the mobile device sends an *acknowledgement* (F) to the wireless carrier's server (A) for the purpose of informing the telecom corporation which mobile web service the mobile user chose. Adjustments are made to the broadcast channel allocations (transmissions between B and C) based on the popularity of each web service over a specified period of time.

Chapter 7

Results Analysis

For the purposes of this experiment, a Java-based simulator was devised which mimics the real world by generating random-sized mobile web services and their associated data. Additionally, it represents the functionality of the proposed framework by, among other things, checking the two thresholds (*i.e.,* Period of time and Pre-defined number of requests/user acknowledgments) to reallocate the channels more efficiently.

7.1 Configuration

The characteristics of the simulated data we utilized for testing purposes are shown in Table 7.1. We ran the same generated data through both the Greedy and Dynamic Programming (DP) allocation approaches in order to make a more fair comparison. The configurations were small in this example to more easily display the result as a tree. However we will provide charts showing the utilization of more complex configurations.

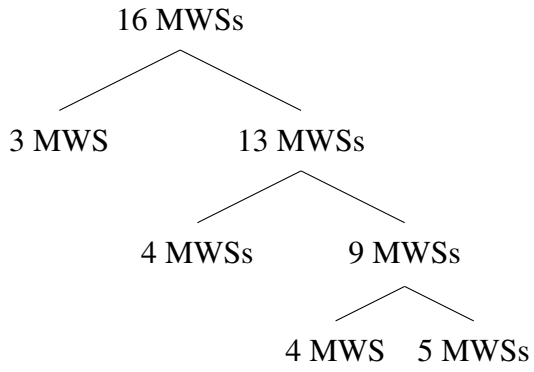
Table 7.1: Configuration Variables

Variable	Value
Number of mobile web services	16
Number of broadcast channels	4
Range of numbers each requests	100-150
Range of mobile web service size & its data	10-15 buckets

As you can see the first testing scenario and results below were based on there being sixteen available mobile web services and four available broadcast channels. Additionally

the threshold for “maximum number of requests” was set at 200 and the size of a mobile web service was given a range of between 3 & 6 buckets. After creating sixteen mobile web services, each was assigned a random number of requests (*user acknowledgements*). Therefore, based on their relative weight (75% on popularity and 25% on size) the algorithm allocates the mobile web services among the four broadcast channels. In the first step, the sixteen MWSs are divided into two groups with one MWS (“highest weighted”) going into channel #1 and the remaining fifteen MWSs being put into channel #2. In subsequent steps the algorithm will continue to “skim off” the “weightiest” MWSs and put them in channels by themselves (see channel #2 has two of them and channel #3 has six). At the end of the allocation cycle, the MWSs with the lowest weights (seven MWSs in this case) are put in the last channel (#4).

1.



In Table 7.2 a representative sample of the results is presented from when we ran the simulation where the configuration was sixteen mobile web services and four broadcast channels. As you can see the Dynamic Programming method produces a slightly better AAT for the mobile users. However, in order to come up with this small improvement, DP performed significantly more calculations than the Greedy algorithm (see Figure 7.1). Therefore, the Greedy algorithm represents the more efficient (*i.e.,* uses fewer resources) approach.

In subsequent iterations of the experiment, one variable (*i.e.,* either the number of MWSs or number of broadcast channels) remained constant while the other’s quantity was variable and chosen from a wide range of values. For instance in one scenario we ran the simulator using a fixed number of broadcast channels (10) and various quantities and sizes of MWSs.

Table 7.2: Results comparison

Allocating method	Average access time
Greedy method	207.16
Dynamic Programming method	159.56

As you can see in figure 7.3 the DP algorithm does come up with lower AAT. However when you look at Figure 7.1 it becomes quite apparent that these the lower AAT figures came only as a result of significantly more calculations having been performed during the process. This same phenomenon occurred when we made the number of MWSs fixed (200) and the number of broadcast channels variable (see figure 7.2).

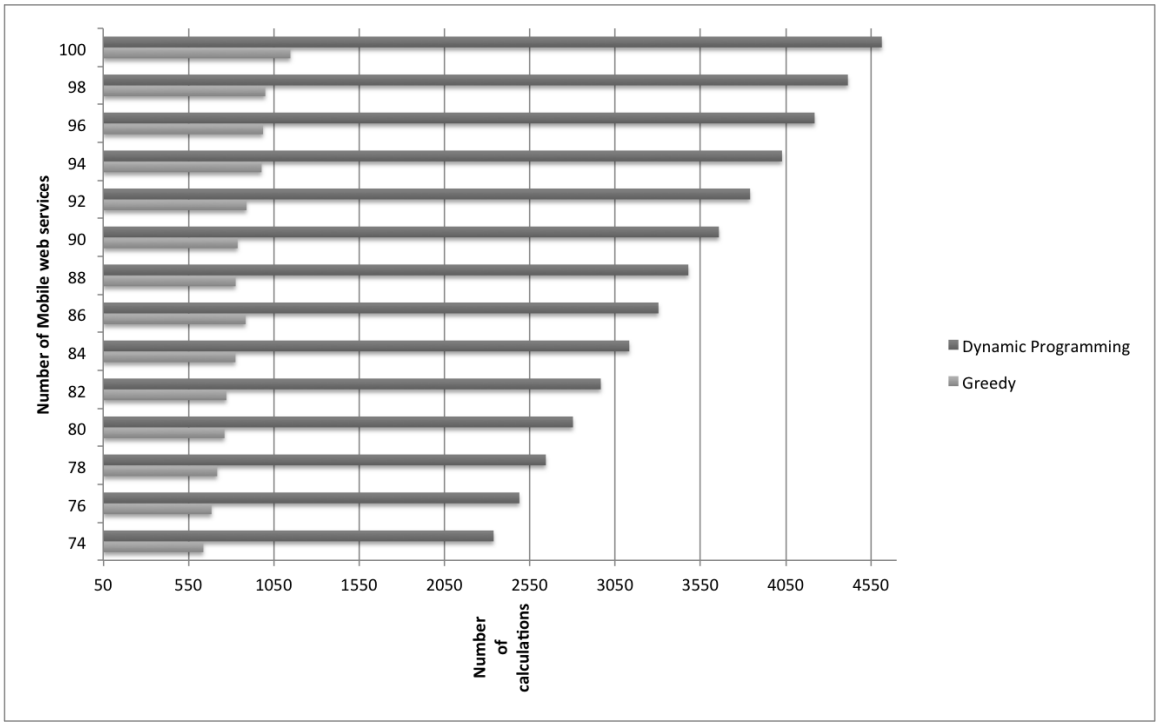


Figure 7.1: Comparison of the number of calculations performed in Greedy vs. Dynamic Programming.

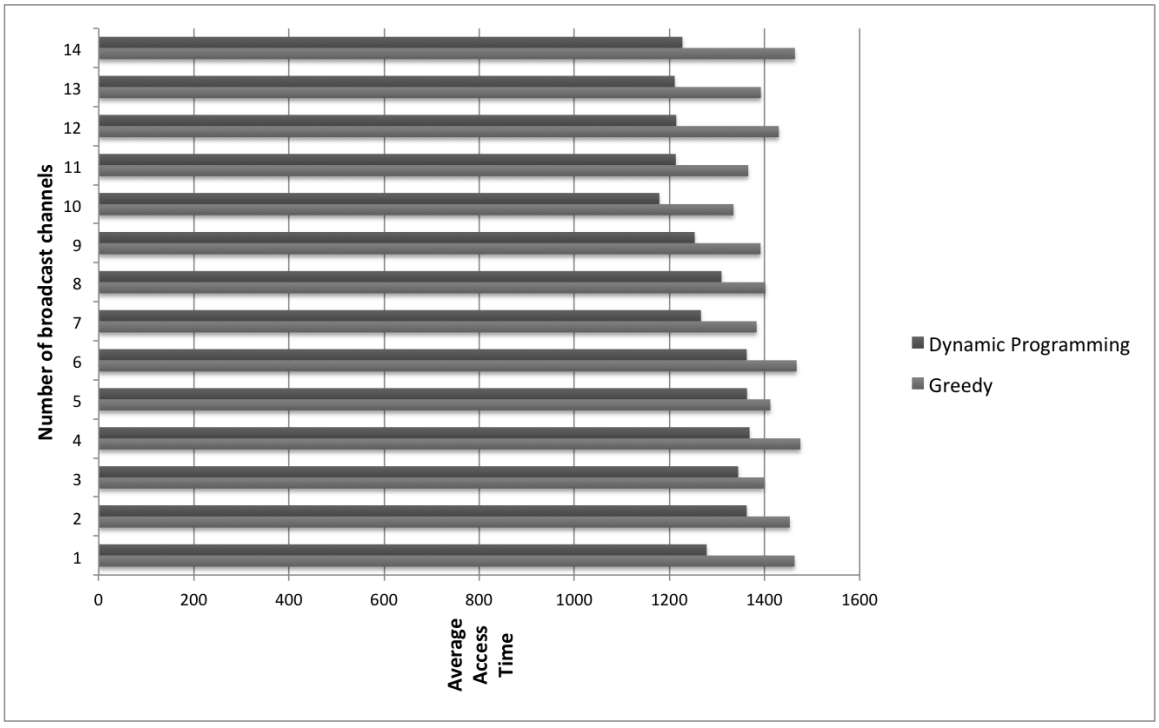


Figure 7.2: A fixed number of mobile web services (200) with a variable number of available broadcast channels.

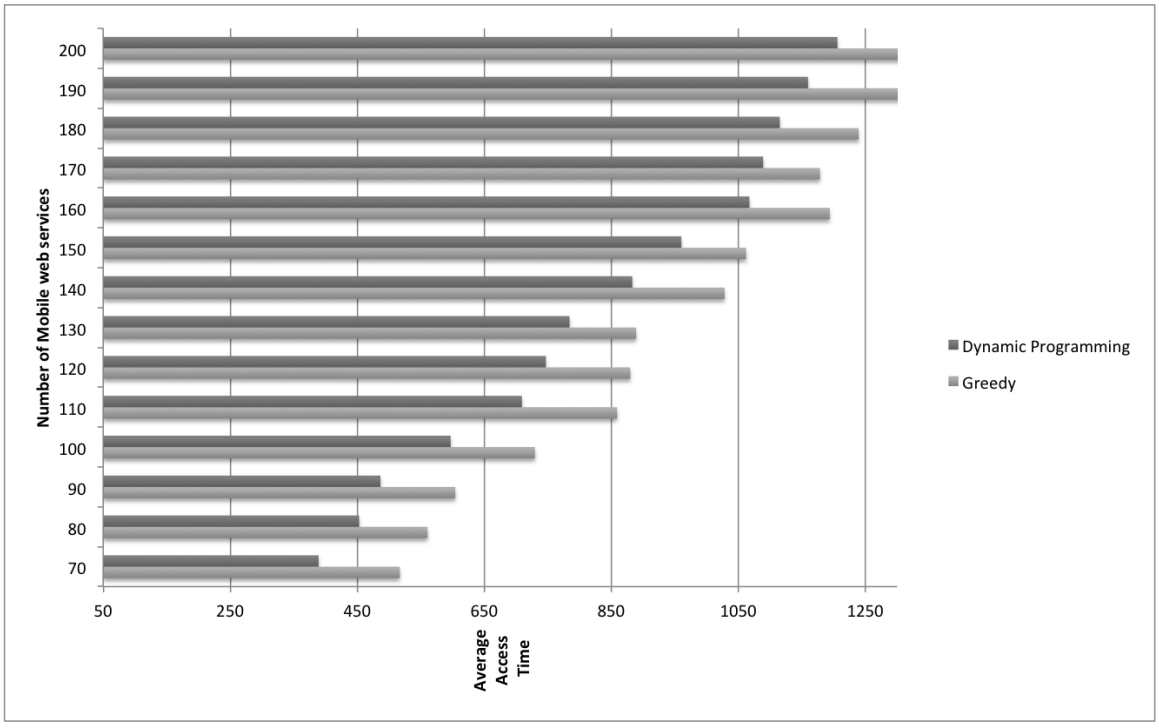


Figure 7.3: A fixed number of broadcast channels (10) with a variable number of available mobile web services.

Chapter 8

Conclusions

From the beginning of this paper, the goal was to improve the mobile device user's experience when accessing the internet by way of their mobile service provider. With the current trend of utilizing broadcast networks to disseminate mobile web services (to unlimited numbers of customers) being adopted by many telecom companies, we included this methodology in our solution. In the interest of providing further efficiencies (*e.g.,* faster response time, decreased usage of the mobile device's limited resources), the process of skewing the mobile web services data allocation to the broadcast channels was designed based on the number of requests received for each of the available services over a set period of time. The system allows for the continuous updating of the allocation schedule to account for fluctuations in demand. As we had hoped, our simulation containing the skewed broadcast channel data allocation arrangement worked with variable sized mobile web services. Finally, we produced nearly optimal average access time (using our Greedy algorithm) without excessive use of the system's limited resources.

8.1 Future Work

Future work would logically involve applying this interaction framework (the efficient channel design and the allocation system) to an actual wireless carrier's broadcast network. In addition, further development needs to be done on the user application so that it would produce real-time statistics which can be received, analyzed, and acted upon by the wireless carrier's servers. It would be expected that further enhancements could be forthcoming based on collaboration with stakeholders in this process (*e.g.,* cell phone manufacturers, web service providers, wireless carriers, *etc.*) Finally, we look forward to receiving any constructive and innovative feedback from other researchers which would bring this "mobile web services allocation through broadcast network" configuration to the next level.

Bibliography

[1] 3GPP2. Cdma2000 high rate broadcast-multicast packet data air interface specificatio. "http://www.3gpp2.org/public_html/specs/C.S0054-0_v1.0_021704.pdf", 2004. [Online; accessed 18-December-2011].

[2] Swarup Acharya, Rafael Alonso, Michael Franklin, and Stanley Zdonik. Broadcast disks: data management for asymmetric communication environments. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, SIGMOD '95, pages 199–210, New York, NY, USA, 1995. ACM.

[3] Fahad Aijaz, Seyed Mohammad Adeli, and Bernhard Walke. Middleware for communication and deployment of time independent mobile web services. In *Proceedings of the 2008 IEEE International Conference on Web Services*, pages 797–800, Washington, DC, USA, 2008. IEEE Computer Society.

[4] Wu Chou and Li Li. Wipdroid - a two-way web services and real-time communication enabled mobile computing platform for distributed services computing. In *Proceedings of the 2008 IEEE International Conference on Services Computing - Volume 2*, pages 205–212, Washington, DC, USA, 2008. IEEE Computer Society.

[5] Telegraph Media Group Limited. Rise in nomophobia: fear of being without a phone. "http://www.telegraph.co.uk/technology/news/9084075/Rise-in-nomophobia-fear-of-being-without-a-phone.html", 2012. [Online; accessed 20-April-2012].

[6] W3C. The world wide web consortium - web services. "http://www.w3.org/TR/ws-gloss/", 2009. [Online; accessed 20-December-2011].

[7] Xu Yang and A. Bouguettaya. Semantic access to multichannel m-services. *Knowledge and Data Engineering, IEEE Transactions on*, 21(2):259 –272, feb. 2009.

[8] Xu Yang and Athman Bouguettaya. Efficient access to wireless web services. In *Proceedings of the 7th International Conference on Mobile Data Management*, MDM '06, pages 30–, Washington, DC, USA, 2006. IEEE Computer Society.

[9] Xu Yang, Athman Bouguettaya, and Xumin Liu. Efficient access to composite m-services. In *Proceedings of the 2009 IEEE International Conference on Web Services*, ICWS '09, pages 381–388, Washington, DC, USA, 2009. IEEE Computer Society.

[10] Xu Yang, Athman Bouguettaya, and Xumin Liu. Semantic-based access to composite mobile services. *Int. J. Web Service Res.*, 8(3):70–100, 2011.

[11] Wai Gen Yee, S.B. Navathe, E. Omiecinski, and C. Jermaine. Efficient data allocation over multiple channels at broadcast servers. *Computers, IEEE Transactions on*, 51(10):1231 – 1236, October 2002.

[12] Wai Gen Yee, Shamkant B. Navathe, Edward Omiecinski, and Chris Jermaine. Bridging the gap between response time and energy-efficiency in broadcast schedule design. In *Proceedings of the 8th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '02, pages 572–589, London, UK, 2002. Springer-Verlag.

[13] Baihua Zheng, Wang-Chien Lee, and Dik Lun Lee. Search k nearest neighbors on air. In *Proceedings of the 4th International Conference on Mobile Data Management*, MDM '03, pages 181–195, London, UK, 2003. Springer-Verlag.