EFFECTS OF COMPRESSION ON SYSTEM THROUGHPUT IN WIRELESS APPLICATION PROTOCOL (WAP) 2.0 ARCHITECTURE.

By

KASHIF KHAN

Masters of Computer Science

Oklahoma State University

Stillwater, Oklahoma

2005

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
In partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2005

EFFECTS OF COMPRESSION ON SYSTEM THROUGHPUT IN WIRELESS APPLICATION PROTOCOL (WAP) 2.0 ARCHITECTURE.

Thesis Approved:

_________________Dr. V. Sarangan_________________
Thesis Advisor

_________________Dr. J. P. Chandler_________________

____________________Dr. N. Park___________________

________________Dr. A. Gordon Emslie______________
Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to thank all my committee members who helped me in completing my thesis. Specially, Dr. Venkatesh Sarangan he was my inspiration, he helped me in every step of the way. He was always there when I needed him. He not only acted as my advisor but he was like a friend. So thank you very much, Dr. Sarangan, for all your guidance and help, you are THE BEST.

I would like to thank all my friends specially Akheer Group who helped me keep up my morale and gave me a lot of good memories. I would like to thank my friend Katherine Taylor. She was always there to help me whenever I needed it.

I would like to thank from the bottom of my heart to my uncle Dr. Azhar Khan, Anti Dr. Khalida Azhar and my cousins Adeel and Nabeel for all their support and help.

I would also like to thank my brother Atif Khan, my sister Sobia Faheem and particularly my beloved parents Mr. Inayat Ullah Khan and Mrs. Rahat Khan. Thank you for all your support, help and prayers. I would not be standing here without you. You are the best parents one can ever ask for.

In the end I would like to thank ALLAH (SWAT) for giving me the patience and guidance that was needed to complete my research.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

LIST OF GRAPHS

CHAPTER 1

INTRODUCTION

1.1 Definition of WAP

“Wireless application protocol (WAP) is an application environment and a set of communication protocols for wireless devices designed to enable access to the Internet and advanced telephony services to the wireless community.” (W@P Forum) [6]

Due to the rapid growth of wireless devices, Internet and consumer demand of Internet access over wireless devices a communication protocol was needed for handheld wireless devices. But most of the available protocols required high bandwidth networks, high speed CPUs with large memory to store data. Also there were a number of different wireless networks that used different standards.

There were numerous differences in terms of device itself e.g.:

- *Form factor:* The size of the wireless device is small, enough to fit in the palm of your hand or in a pocket [12].
- *CPU:* CPU of the wireless device is small and less powerful as compared to the CPUs of the desktops [12].
- *Memory and storage:* Storage capacity of the wireless device is very limited [12].
- *Battery:* Since wireless devices use battery power, CPU cannot use excessive battery power for computation [12].

- *Display:* Display of wireless devices is very small, has limited resolution and color capabilities compared to regular desktops [12].
- *Input:* Wireless device have a very different way for input of data [12]. These do not have a regular keyboard and have to use a small keypad to enter data.

To handle the above mentioned issues wireless application protocol (WAP) was developed. WAP developed a wireless application environment (WAE) and virtual machine (VM) for mobile devices and a protocol stack for the wireless network. All these components are defined as follows:

Micro Browser: This browser is a part of the WAE and has the capability to render data according to the size of the screen [12]. This browser puts more emphases on functionality than rendering of contents. Browser selects an appropriate way to render pages on the mobile device.

Virtual Machine: This is used to execute the scripting language for the micro browser [12]. This VM is well suited for the memory and CPU constraints of the mobile device.

Protocol Stack: This protocol stack is designed to take into account the bandwidth limitations and reliability issues [12]. It operates on IP networks and uses User Datagram Protocol over IP wherever possible. It can also operate on non IP networks as well. The content to be transmitted is encoded and compressed to reduce the amount of data to be transferred over the wireless network.

Wireless Markup Language (WML): HTML is not suited for the wireless environment as it strongly concentrates on rendering of the document [12]. Also HTML header size is considerable. So a markup language was developed for the wireless environment called WML. WML is derived from XML and contains elements that are easy to render on the mobile device. Also there is a scripting language called WMLScript for mobile devices to write any WAP Client side code.

1.2 WAP Architecture

Let us look at the World Wide Web model. It will help us in understanding the Wireless Application Protocol Model.

1.2.1 World Wide Web (WWW) model:

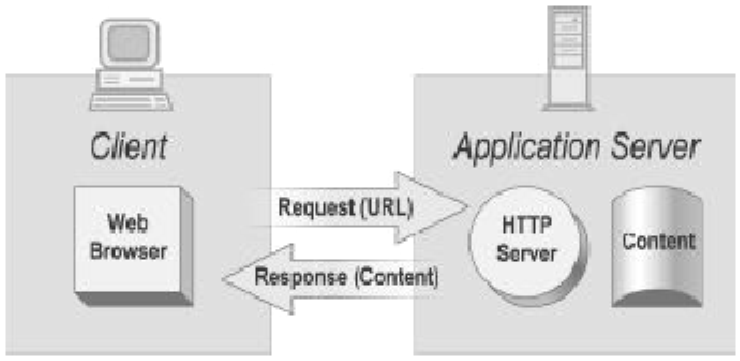


Figure 1.1: WWW Programming Model

WWW architecture shown in fig 1.1 is a very powerful programming model [6]. In this programming model, applications and content use standard data formats. These contents and applications are displayed using a web browser. Web browser is a networked application used to send requests to the network WAP Server. The network WAP Server sends back the requested data using the standard HTTP format, which is then displayed by the web browser.

The standards defined by the WWW are:

- Standard naming model: Internet standard uniform resource locator (URL) is used to name the WAP Servers and the content on (WWW) [6].

- Content type: A specific content type is assigned to all the contents on WWW [6]. This is done so that the browser can process the contents based on their type.
- Standard content formats: Some of the standard formats supported by the browser are HTML, JavaScript, and VBScript etc [6].
- Standard Protocols: Protocols are used by the browsers to communicate with the web WAP Server [6]. Hyper text transfer protocol (HTTP) is commonly used for communication on WWW.

1.2.2 WAP Model:

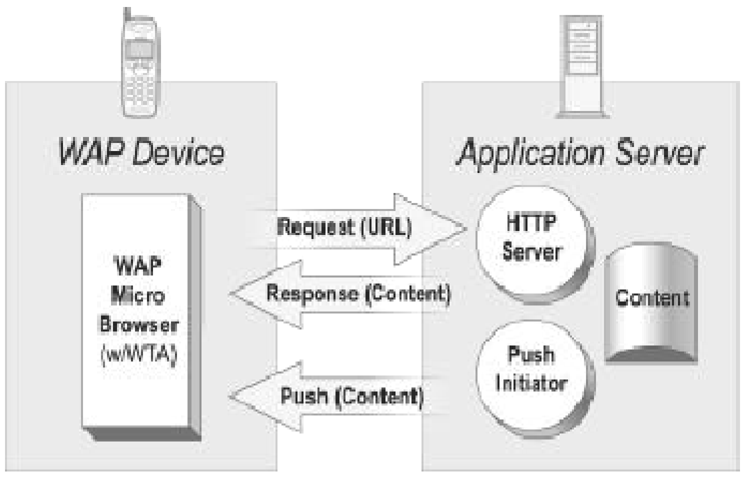


Figure 1.2: WAP Programming Model [6]

WAP programming model as shown in fig. 1.2 has adapted the WWW programming model [6]. However some enhancements have been made in the model to make it more suitable for the wireless environment. Some of the major enhancements in the WAP model are:

- Push Mechanism, and
- Telephony support (WAT)

In WAP the well known request-response mechanism is referred to as "Pull" mechanism [6]. In Push mechanism the content is delivered to the WAP Client without any WAP Client request [10]. Telephony support refers to the regular telephonic functions like receiving a telephone call.

Content format for WAP is the same as that of the WWW content format. Also the content is transported using standard communication protocols based on WWW communication protocol. WAP micro browser provides the user interface and is analogous to the standard web browser. Some of the standards defined by the WAP are:

- Standard naming model: WWW standard URLs are used to get WAP specific data from the origin WAP Server and to identify local resources in the WAP Client (device) e.g. address book, calendar etc [6].
- Content type: WAP contents types are consistent with the WWW types [6]. This is done so that the browser can process contents based on their type.
- Standard content formats: These formats are based on WWW technology and include calendar information, images and scripting languages etc [6].
- Standard Protocols: WAP communication protocols are used by the mobile browser to communicate with the web WAP Server [6].

1.3 WAP Clients

Mobile cell phones, PDA's and other hand held devices capable of accepting Internet content are WAP Clients. The capabilities of the WAP Client are limited because of the small screen size, less battery and computational power, and low resolution.

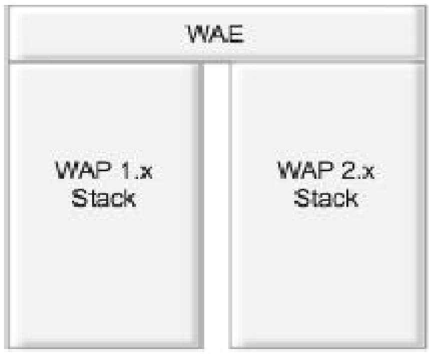


Figure 1.3: WAP Device Dual Stack Support [6]

WAP Clients use two types of protocol stack (shown in fig. 1.3 above) to support both WAP 1.x and WAP 2.0 [6]. WAP 2.0 uses the wireless profiles HTTP (WPHTTP) and Wireless profiled TCP (WPTCP) to communicate with the WAP Client over the wireless region of the Internet. This WPHTTP and WPTCP are lighter than the HTTP, TCP used for the wired Internet.

1.4 WAP Servers

WAP Servers are the regular WAP Servers that store HTTP pages. Two types of pages are stored in the WAP Server HTTP pages and WML (wireless markup language) pages. If WAP Client is using WAP 1.x then the WAP Server sends WML page if available, otherwise WAP Server will send HTTP page and WAP Gateway will convert the HTTP page into WML page before sending it to the WAP Client. If the WAP Client uses WAP 2.0 to make a request then regular HTTP pages are sent to the WAP Client. The WAP Gateway in this case converts the HTTP protocol to WPHTTP and TCP to WPTCP before sending it to the WAP Client.

1.5 WAP Gateways (Proxies)

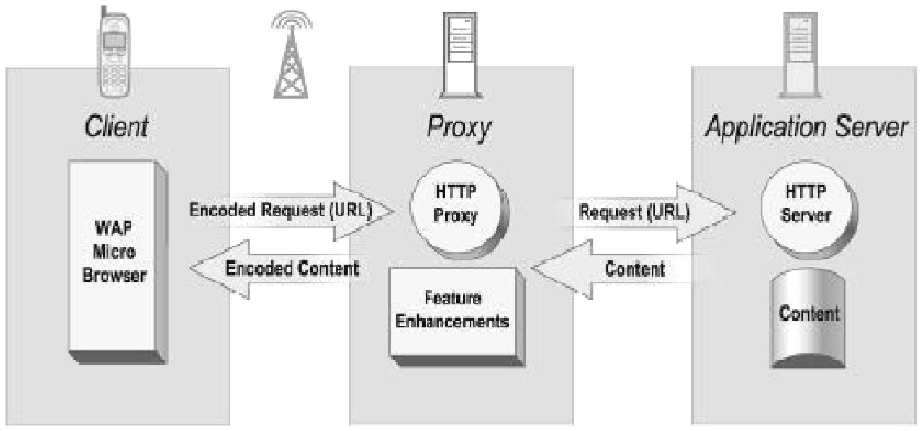


Figure 1.4 Performance Enhancing Gateway/Proxy [6]

Gateway is an intermediate program that acts both as a WAP Server and WAP Client as shown in fig 1.4. WAP Gateway makes requests on behalf of other WAP Clients [12] [3]. Proxy typically resides between WAP Clients and WAP Servers that have no means of direct communication e.g. across firewalls.

Proxies are used to enhance the connection between the wireless domain and WWW. Following are some of the functionalities provided by proxy:

Encoders and Decoders: These are used when WAP 1.x is being used [6]. This reduces the size of the content to be delivered. This reduction in size helps in the effective utilization of underlying wireless environment.

Gateway Protocol: This converts the wireless protocol stack to WWW protocol stack and vice versa. Gateway also performs DNS lookups for WAP Servers named by the WAP Client [6].

Caching Proxy: This is used to store the frequently used resources thus increases the performance and network utilization [6].

User Agent Profile Management: This contains the WAP Client preferences and WAP Client capabilities. This helps WAP Servers to sending WAP Client compatible data and applications [6] [13].

This infrastructure allows developers to develop application and services using WWW technologies that are compatible with the WAP Client environment.

1.6 Supporting WAP Servers

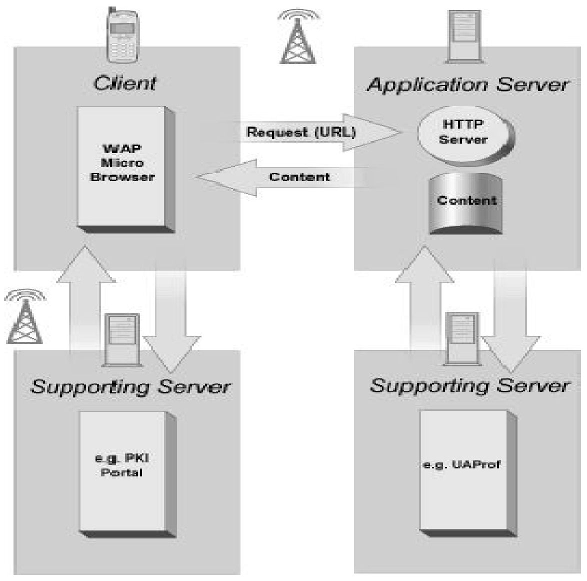


Figure 1.5 Supporting Services [6]

Supporting WAP Servers (shown in fig 1.5) are used to provide different kinds of services [6]. Some examples of supporting WAP Servers are as follows:

PKI Portal: This helps devices in creating new public key certificates [6].

UAProf WAP Server: This helps applications in retrieving WAP Client capabilities and personal profiles of users [6].

1.7 WAP Network Elements

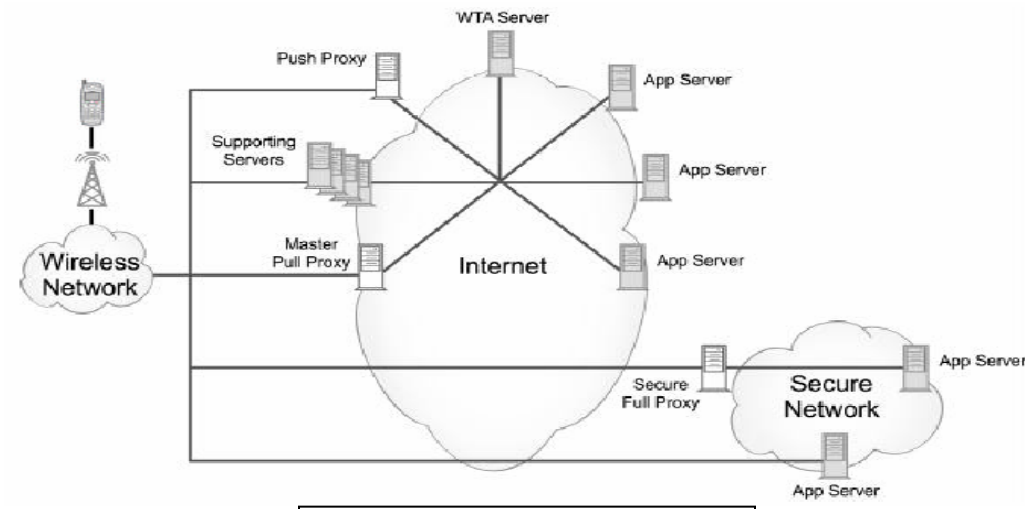


Figure 1.6 WAP Network [6]

The most common WAP architecture consists of three parts:

1- WAP Client

2- WAP Proxy WAP Server or WAP Gateway

3- Web WAP Server

Other configurations are also available as shown in fig. 1.6 that can be used for the communication between the mobile WAP Client and the web WAP Server [6]. WAP utilizes the proxy selection configuration to select the most appropriate proxy for communication it can also communicates directly with the WAP Server.

Proxies can be located in the wireless carriers or independent service providers [6]. Proxies help in optimizing the communication and provide feature enhancements related to wireless network e.g. telephony and location etc. The device can also communicate directly with the application WAP Server to provide secure connection between the mobile device and the WAP Server.

1.8 WAP 1.x protocol Stack

The WAP 1.x architecture is shown in the fig 1.7 below.

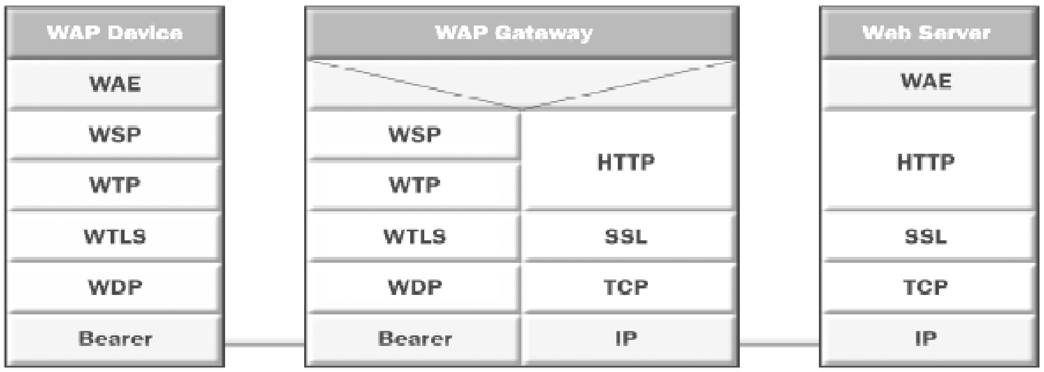


Figure 1.7: WAP 1.x Architecture [6]

“WAP uses protocols that are complete binary-based protocols and work on top of the common Internet User Datagram Protocol (UDP/IP), so the usual Internet protocols HTTP/IP are not used.” [16]

Direct mapping of WAP protocol stack onto WWW protocol stack is not possible. WWW protocol architecture consists of seven layers where as Wireless protocol architecture consists of five layers but some comparison is possible as shown in fig 1.8.

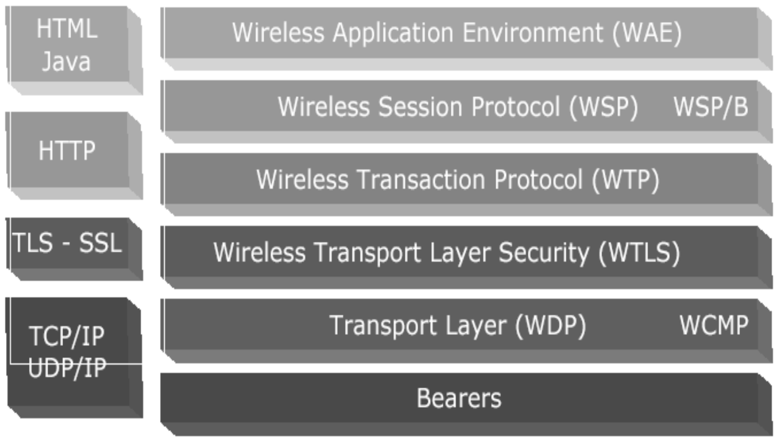

HTML
Java
HTTP
TLS - SSL
TCP/IP
UDP/IP
Wireless Application Environment (WAE)
Wireless Session Protocol (WSP)
WSP/B
Wireless Transaction Protocol (WTP)
Wireless Transport Layer Security (WTLS)
Transport Layer (WDP)
WCMP
Bearers

Figure1.8 WAP Protocol Stack (WAP Overview) [12]

The functionality of HTML and Java are incorporated by the WAE and the WSP layers of the WAP protocol. Hyper text transfer protocol (HTTP) functionality is handled by the WSP and the WTP layers. The security feature provided by transport layer security

(TLS) is provided by WTLS layer in WAP. For transportation over the wired network either TCP or UDP over IP is used. For wireless networks UDP over IP is used; alternatively if the IP is not supported by the network layer then WDP is used for transportation. Fig. 1.8 shows the different layers in WAP 1.x architecture. Let us discuss each layer briefly:

Wireless Application Environment (WAE): This layer defines the format of contents and services for the application environment of mobile devices [12] [5]. WAE presumes the existence of a user agent for interpreting the contents referenced by the URL. Contents handled by the user agents are:

WML: This is handled by the micro browser.

WML Script: This is executed by the virtual machine.

Some of the formats supported by this layer are vCard, vCalendar for transferring phone book and calendar information between applications and WBMP for wireless bit map.

Wireless telephony API enables the user to interact with the device, and the services and facilities on the operator's network. For example access to a phone book on mobile phone.

Wireless Session Protocol (WSP): This layer is used to create secession between the WAP Client (mobile device) and the WAP Server [12] [5]. This layer is responsible for:

- Suspend and resume sessions: This is an important feature as the connection may drop and need to be reconnected so that it can pick up from where it left off.
- Long lived sessions: This is necessary as there might be huge delay in the wireless environment.

- Capability negotiation: Since capabilities of mobiles vary, so the WAP Server can not treat all mobiles the same way. Capability negotiation helps in customizing the session between WAP Client and the WAP Server.

Both connectionless and connection oriented sessions are supported by WSP. For connectionless session no session setup is needed. Connection oriented session runs over TCP where as the connectionless session runs over WDP.

There are some browsing extensions of WSP which help in making the protocol more compatible with the environment it is going to be used in. Semantics of these extensions are based on HTTP 1.1. It helps in the transmission of session and content headers. It also encodes the content headers to make them more efficient for over the air transmission.

Character sets, language and the user agent profile are defined by the Content headers. Theses properties define the characteristics of the device. Push capabilities are supported by the session layer.

Wireless Transport Protocol (WTP): It consists of the typical request/response mechanism [12] [5]. Some of the facilities provided by this layer are:

- Retransmission and selective retransmission of lost or corrupted message.
- Message concatenation etc.

It also provides an abort capability to indicate to the WAP Server that the results of any processing are not required anymore. There are three classes of transportation:

- Class 0: This provides an unreliable stateless connection and it does not support the abort function.

- Class 1: This provides a reliable connection and maintains the state of the communication. It supports abort and retransmission but does not support the result functionality.
- Class 2: This is similar to class 1 but it also supports the result functionality. “Hold on” reply result is also present, indicating that the WAP Server is busy processing the request.

Wireless Transport Layer Security (WTLS): It is similar to TLS. It provides over the air secure data transmission [14] [5]. It also supports functions like:

- Message integrity.
- Message authentication.
- Message privacy.
- Key exchange mechanisms.
- Signature functions.
- Symmetric and asymmetric encryption ciphers.

Also, WTLS can be integrated with the Public Key Infrastructure (PKI).

Transport Layer: The transport layer uses two types of protocols, UDP over IP and WDP. WDP is used on wireless networks for connectionless non reliable transmission. This protocol supports message segmentation and reassembly, UDP based port numbers are also supported by this WDP protocol [14] [5].

Bearers: There are a number of over-the-air bearers available that can be used to communicate with the mobile devices. Some of the bearers are GSM, IS-136, CDMA etc [14] [5].

1.9 WAP 2.0 protocol Stack

The WAP 2.0 protocol stack is the same as used by the WWW for wired Internet as shown in fig. 1.9 below [16]. The only difference is that on the WAP Server side we use the HTTP/TCP/IP protocol stack while on the WAP Client side WPHTTP/WPTCP/IP is used. The only difference between these two protocols is that the WAP Client side protocol stack is optimized for wireless networks. The conversion of the HTTP/TCP/IP to WPHTTP/WPTCP/IP is not a complex process as compared to the conversion of HTTP protocol to WAP protocol. Let us discus the protocol stack used by WAP 2.0.

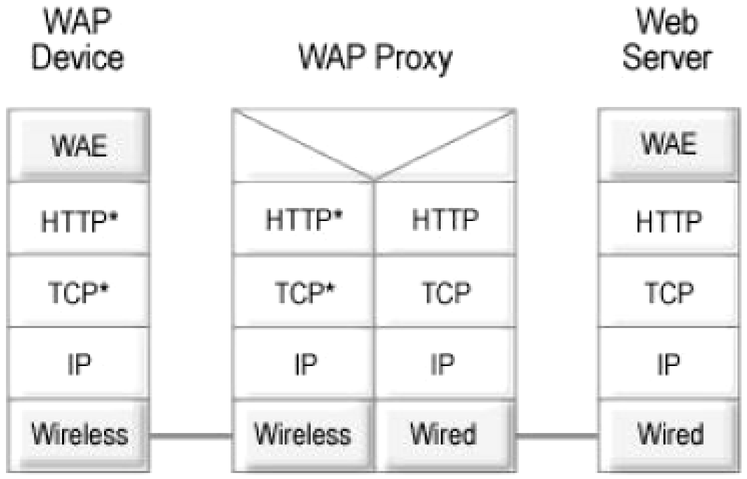


Figure 1.9 Wireless Profiled HTTP and TCP with WAP Proxy [16]

WAE or Application Layer: This is the layer where the request for data and services are processed. Applications like HTTP and FTP etc. listen at their respective ports for a request to process. Then, these applications interact with the word processor or the web browser etc. according to the needs of the request [15].

HTTP Layer: HTTP is a set of rules for exchanging files on the Internet. This protocol is used by the web browser to surf the Internet. The WAP Client makes HTTP request by using the web browser. The web browser formats the WAP Client’s request into HTTP/TCP/IP request and sends it to the WAP Server. HTTP application running on the

WAP Server is listening for the request. It responds back by sending the requested page or a file back to the requestor [15].

TCP Layer: This protocol is used to connect the WAP Client and the WAP Server to each other. This is a connection oriented protocol so it means that the WAP Client and WAP Server know about each other. This protocol is used to determine hoe to communicate, where to send data and how the data will be received. This protocol guarantees the delivery of packets by sending acknowledgments after each packet is received [15].

IP Layer: This protocol is responsible for determining the IP address of the source (WAP Client) and the destination (WAP Server) of every packet. The IP address is the logical address that is assigned to the WAP Clients and the WAP Servers by the network administrator. Each WAP Client and WAP Server has a unique IP address [15].

1.9.1 WAP 2.0:

WAP 2.0 is a significant development in the field of wireless communication. WAP 2.0 allows the transfer of HTTP pages over the wireless and it also supports WAP 1.0 architecture for backward compatibility. WAP 2.0 architecture uses Wireless profiled HTTP and Wireless Profiled TCP to deliver content over the wireless region. The wireless profiled versions are interoperable with TCP and HTTP.

HTTP 1.1 can be used for communication between the WAP Client and WAP Server so WAP 2.0 does not require a WAP Proxy [1]. However, WAP proxy can be used to enhance the performance by optimizing the communication process. It may also offer location, privacy, and presence based services. Also, WAP proxy is necessary for Push functionality.

The WAP Gateway is located by using the wireless profile TCP. Also, Wireless profiled HTTP is used to further improve the performance.

The basic model of interaction between the WAP Client, WAP proxy and WAP Server is the regular request response model used by WWW model. The WAP Client should have the capability to interact with WAP HTTP proxies and origin WAP Servers. This transfer layer provides a service access point with may be used by the "Pull" and "Push" data transfer models [9]. Pull is achieved by using the request-response model from HTTP. Push is achieved by considering the WAP terminal as the WAP Server. This helps us in modeling "Push" as a request-response towards the WAP terminal. The push architecture is shown in fig. 1.10.

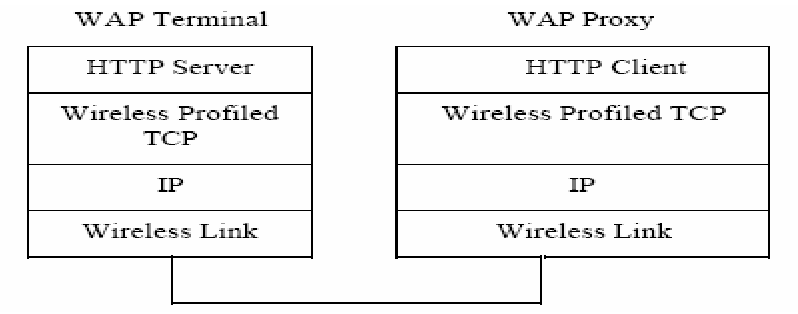


Figure 1.10 Wireless Profiled HTTP And TCP for WAP Push [6]

The existence of WAP proxy allows the use of split TCP connection [8]. The connection between WAP Client and WAP proxy is established by using WPTCP and the connection between WAP proxy and WAP Server is established by using regular TCP. Hence the use of a proxy allows is to optimize the transaction between the WAP Client and the WAP Server. The WPTCP can also be used for end-to-end connectivity i.e. without any TCP connection information or intermediate nodes.

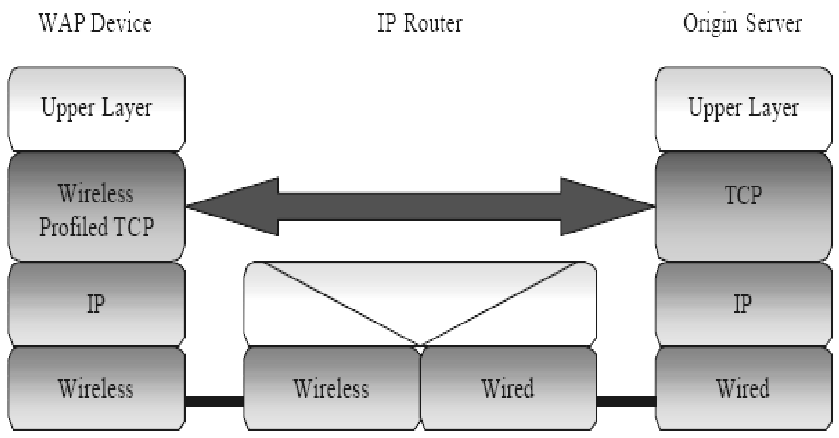


Figure 1.11: Wireless Profiled TCP without WAP Proxy [8]

WPHTTP supports message body compression of responses and allows the establishment of a Tunnel using the CONNECT method. The existence of WAP Gateway (WAP Proxy) is not mandatory as shown in fig. 1.11. It is possible to communicate between the WAP Client and WAP Server using HTTP. WAP Gateway optimizes the communication between WAP Client and WAP Server. This optimization is achieved by using WPHTTP/WPTCP for communication between the WAP Gateway and WAP Client [13] and by using HTTP/TCP between the WAP Gateway and WAP Server as it is optimized for use over wired networks and vice versa. The use of WAP Gateway can also provide services such as location, privacy, and presence based services [1].

CHAPTER 2

WORK DONE BY OTHERS ON WAP ARCHITECTURE

2.1 Enhancements in WAP 1.x Architecture

Horizontally WAP 1.x architecture consists of the WAP Client, WAP Gateway, and the WAP Server [2]. WAP Client uses the WAP protocol stack for its transactions with the WAP Server. The WAP Gateway uses two protocol stacks the WAP and the HTTP protocol stack. WAP Gateway works as a translator between the WAP Client and the WAP Server. It converts the WSP request to HTTP request before sending it to the WAP Server and vise versa. WAP Server, which can be a usual WWW WAP Server, uses HTTP protocol to communicate with the WAP Client. Thus, WAP Gateway is used to translate the protocol from HTTP to WAP protocol.

The language used for this environment is WML and WMLScript. WML is an XML based markup language [7]. WML is used to define cards and decks which are like HTML pages. These cards and decks suitable for the environment of the handheld mobile devices as the screens of those devices can not accommodate the regular HTML page. The handheld mobile devices use a micro browser to display WML contents. The micro browsers also run applications written WMLScript which is similar to JavaScript. The WML and WMLScript applications are converted into byte code before they are sent to the WAP Gateway or vise versa. The byte code of the WML and WMLScript makes it suitable for the wireless network. Also the headers and their values of the WSP protocol are coded into binary format. Each header and their values are given a distinct binary code. This binary format helps in reducing the size of the headers for transfer over the wireless network.

One of the improvements that were made on this architecture was the introduction of compression algorithms. These compression algorithms were used to compress byte code generated by the WML (Script) [2]. Thus, it further reduced the amount of data to be sent over the wireless network.

Problem with this approach was that its effectiveness depended on file size and the network bearer. Compression algorithms were not effective for small file sizes. In fact, compression decreased the overall performance if the file size was small enough to fit one data packet with out compression [2]. For example, if the file size was small enough to fit one data packet and if compression was applied on such files then, these files have to be decompressed at the WAP Client side. Thus, increasing the time needed to deliver data to the WAP Client. In such a case the application of compression algorithms is not recommended.

2.2 Enhancements in WAP 2.0 Architecture

Adopting the most recent standards and protocols, WAP 2.0 has brought the wireless world closer to the Internet [1]. WAP 2.0 architecture is shown below in fig. 2.0. A request by the WAP Client is sent to the WAP Gateway by using the WPHTTP and the WAP Gateway communicates with the WAP Server using the HTTP protocol stack. The development of WAP 2.0 was encouraged by the availability of high speed data networks

(CDMA 2000, GPRS etc.) and current handheld devices, which are far more sophisticated than before.

There are also disadvantages of using HTTP/TCP/IP which are as follows [1]:

- More bits are transferred using HTTP than WSP.
- TCP requires more transactions than WTP.
- Header size is a lot bigger than the header size of WAP protocol.
- Transmitted packets are much larger than the byte encoded ones in WAP1.x.

Improvement in WAP 2.0 architecture is done by reducing the amount of data transferred over the wireless and wired network. This is done by introducing compression algorithms in the WAP 2.0 protocol stack (shown in figure below).

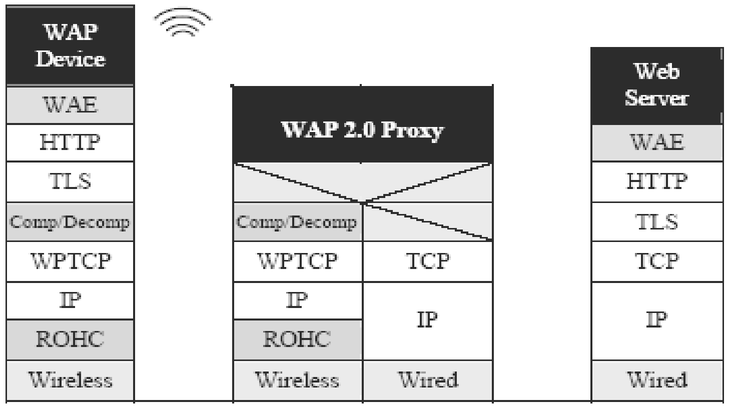
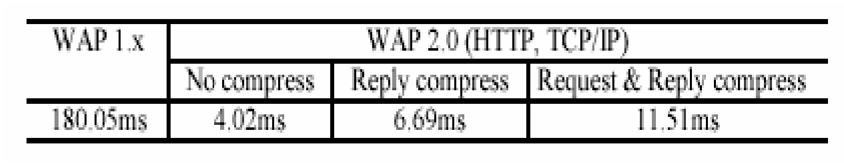


Figure 2.0: WAP 2.0 Architecture with Compression/Decompression [1]

This compression scheme allows TCP content compression along with Robust Header Compression (ROHC) [1]. This combination not only compresses the data but also overcomes the end to end security problem in WAP 1.x. This architecture uses TCP to connect to the WAP Server and WPTCP to connect to the WAP Client. Thus by TLS tunneling security is guaranteed [17].

Above the WPTCP layer content compression/decompression is introduced. Thus, it compresses the HTTP content and HTTP headers both. The original WAP 2.0 architecture offers the compression at the WPHTTP layer, compressing the HTTP content only. After applying the compression at the WPTCP layer ROHC is applied below the IP layer to compress the TCP and IP headers. This will result in further reduction of packet size and will help in conserving the bandwidth over the air interface.

The following table shows the result of employing the above mentioned ROHC technique.



WAP 1.x	WAP 2.0 (HTTP, TCP/IP)		
	No compress	Reply compress	Request & Reply compress
180.05ms	4.02ms	6.69ms	11.51ms

Table 1: Data Processing Delays [1]

CHAPTER 3

PROPOSED CHANGES IN WAP 2.0 ARCHITECTURE

Above mentioned ROHC scheme reduces the packet size thus improves the overall throughput. Also it uses the HTTP/TCP and WPHTTP/WPTCP over the wired and wireless networks respectively. This helps in delivering the regular HTTP pages to the handheld wireless devices.

The basic idea behind my proposed architecture is based on the fact that reducing the size of data packet over the wired and wireless network will increase throughput. ROHC scheme discussed in the previous section compresses/decompresses data for communication between the WAP Client and WAP Gateway. But there is no data or header compression for communication that takes place over the wired network. Thus the possibility of further performance enhancement still exists in WAP 2.0 architecture.

Fig. 3.0 and fig 3.1 gives the general idea of my proposed architecture. As shown in fig. 3.0 compression/decompression and the ROHC have been moved from the WAP Gateway to the WAP Server. By adopting this architecture we can send compressed data over the wired and wireless Internet networks. This will improve the throughput by sending more data in compressed form in lesser time from the WAP Server to the WAP Client.

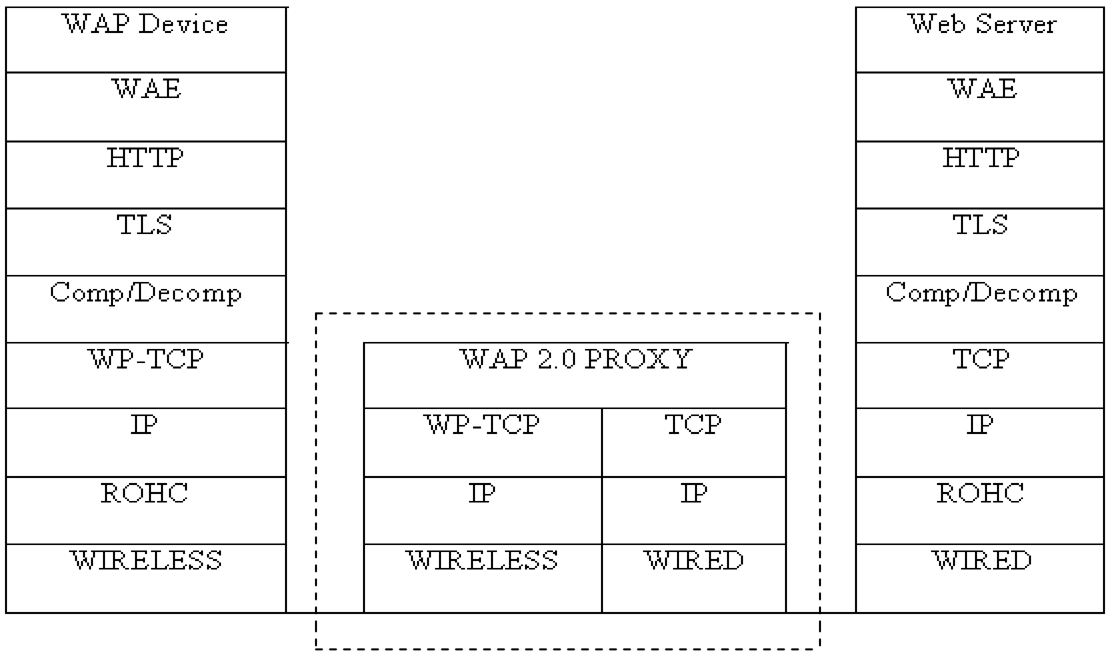


Figure 3.0 Proposed WAP 2.0 Architecture with WAP Gateway

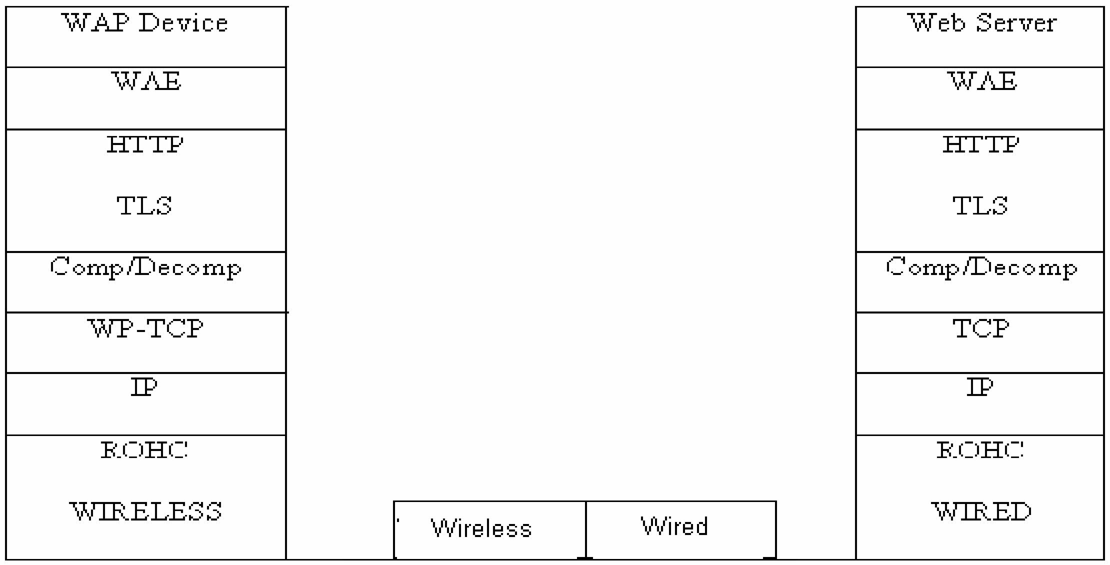


Figure 3.1 Proposed WAP 2.0 Architecture without WAP Gateway

When the WAP Client makes a request, this request is compressed above the WPTCP layer and ROHC is applied on the Request Packet after the IP layer. This compressed request packet is then delivered to the WAP Proxy using WPTCP/IP. The

WAP proxy sends the request to the WAP Server using TCP/IP. When the request reaches the WAP Server it is decompressed and the required information is retrieved from WAP Server.

The reply from WAP Server is again compressed above TCP layer and then ROHC is applied after the IP layer. The reply message is then sent to the WAP Proxy using TCP/IP protocols and then sent to the WAP Client using WPTCP/IP protocols. The reply messages can be stored in the WAP Gateway cache in compressed/uncompressed form. So, if there is a packet loss or some error in the transfer of reply message between WAP Gateway and WAP Client, WAP Gateway can retransmit the lost packet. Also, if the frequently accessed pages are stored in the WAP Gateway cache, WAP Client request can retrieve those pages from the WAP Gateway. This will decrease the response time. There can also be direct communication between the WAP Client and WAP Server in the absence of WAP Gateway. This architecture is shown in fig. 3.1 above.

Since the Reply Message is compressed it will take less time to reach the WAP Gateway and then the WAP Client. Also it will use less network bandwidth of both wireless and wired networks. Hence, more users can be entertained on the same network bandwidth. The benefits of using this architecture are:

1- Improved response time.

2- Less bytes to transfer. This means smaller charges for customers as customers are charged based on the amount of bytes received.

3- Service providers benefit from this architecture by being able to accommodate more WAP Clients without having to expand the network bandwidth.

The response time of this architecture can be further reduced by deploying more and better hardware at WAP Server side and storing the HTTP pages in compressed format.

The compression algorithms compress headers and header values of HTTP/TCP thus, reducing the overall packet size even further.

We will simulate this proposed architecture and study the different scenarios related to it as follows:

1- The difference in performance using the previous and the proposed architecture as discussed in section 2.1 and 3.0 respectively. This will help us in quantifying the efficiency of proposed architecture.

2- The performance improvement (if any) by caching pages accessed by the WAP Client at the WAP Gateway in compressed and uncompressed format. The idea of having cache is to store frequently accessed pages. But the presence of cache also means more work for the WAP Gateway. This has its pros and cons. Our study will help us analyze different cache hit scenarios.

3- Tradeoffs involved in spending more time in compressing/decompressing data in order to reduce transmission times. Compressing/Decompressing data packets has its advantages but it also adds load on the WAP Client and WAP Server, thus increasing the request turn around time. Our study will try to help us understand the tradeoffs involved in this proposed architecture.

4- The performance of the proposed and original architectures for different packet loss probabilities. Our study will show how both architectures perform under various packet loss probabilities.

5- The threshold value of data size at which compression will have more effect on the throughput.

6- Effects of compression/decompression on battery power of handheld devices.

CHAPTER 4

ANALYTICAL MODELING

- Claim "1" and "4" can be shown to provide better results than the original architecture by implementing the equations derived below:

Proposed Architecture Response Time:

Client Request time can be divided as follows,

Time to construct WAP Request at WAP Client side = T_c

Time to send WAP Request over wireless to the Gateway = $T_{wireless}$

Time taken to send WAP Request over the wired network from WAP Gateway to the WAP Server = $T_{wired\text{-}Request}$

Time taken at the WAP Server = T_s

So,

Time Taken by WAP Request to go from WAP Client to WAP Server ($R_{c\text{-}s}$) =

$$T_c + T_{wireless} + T_{wired\text{-}Request} + T_s \text{ ------------- } (1)$$

Each component of this equation is explained below.

T_c = Regular time to construct request + Time for compression after TLS Layer + Time for ROHC compression.

= $\text{Constant}_c + T_{comp\text{-}c} + T_{ROHC\text{-}c}$ -- (2)

$T_{wireless}$ = Time to deliver the Frames over the wireless network

Let, t_1 = Time taken for WAP Request to reach WAP Gateway without error

$$t_1 = \frac{\text{Message Size}}{\text{Wireless Link Speed}}$$

ε_1 = Error drop rate in wireless network

Because of the error drop rate:

Probability of successful transfer over wireless network = $(1-\varepsilon_1)$

This means that in time t_1 some request packets might not get transferred.

So,

Time to deliver all the request packets over wireless network = $\frac{N_{c\text{-}g}\, t1}{1-\varepsilon_1}$ ----------- (3)

Where $N_{c\text{-}g}$ = Number of WAP Request packets sent from WAP Client to WAP Gateway.

$T_{wired\text{-}Request}$ = Time spent at "k" HOPS + Time taken by WAP Request to go from WAP Gateway to WAP Server excluding time spent at "k" HOPS.

$T_{wired\text{-}Request} = T_{HOP} + T_{g\text{-}s}$

Where,

Time Spent at "k" HOPS (T_{HOP}) = [Time spent at one HOP + Time to decompress ROCH at the HOP + Time to compress ROHC at the HOP] x k

We know that Minimum Router (HOP) Transit Time = $0.0213 \times L + 25 \times 10^{-6}$ Sec. [24]

Where, "L" is the Packet Size in bytes.

Time Spent at "k" HOPS (T_{HOP}) = $[(0.0213 \times L + 25) \times 10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] \times k$ --- (4)

Now let,

t_2 = Time taken for WAP Request to go from one HOP to another over the wired network without error.

$t_2 = \frac{\text{Message Size}}{\text{Wired Link Speed}}$

Then,

Time taken by Request to reach WAP server over “k” HOPs = kt_2

Because of bit error rate “ε_2” of wired network

Probability of successful transfer over the Wired Network = $(1- \varepsilon_2)$

Now we derive the successful transmissions over “k” HOPs as follows:

Example:

Let

Number of hops = 2

Drop error rate between each HOP over the Wired Network = ε_2

Then

Probability of Error over two HOPs = $P_e = \varepsilon_2 \text{ x } \varepsilon_2 + (1-\varepsilon_2) \text{ x } \varepsilon_2 + (1-\varepsilon_2) \text{ x } \varepsilon_2$

$= \varepsilon_2^2 + \varepsilon_2 - \varepsilon_2^2 + \varepsilon_2 - \varepsilon_2^2$

$= 2\varepsilon_2 - \varepsilon_2^2$

Now,

Probability of Success over two HOPs = $1- P_e$

$= 1 - 2\varepsilon_2 + \varepsilon_2^2$

$= (1- \varepsilon_2)^2$

Similarly,

Probability of Success over “k” HOPs = $(1- \varepsilon_2)^k$

So,

Time taken by WAP Request to go from WAP Gateway to WAP Server excluding time spent at “k” HOPS($T_{g\text{-}s}$)= kt_2 -- (5)

So,

$T_{wired\text{-}Request} = [(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] \text{ x } k + kt_2$

$= ([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] + t_2) \text{ x } k$

Because of bit error rate “ε_2” of wired network

$$T_{wired\text{-}Request} = \frac{([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] + t_2) \text{ x } k}{(1 - \varepsilon_2)^k}$$

Now for $N_{g\text{-}s}$ number of requests. Where,

$N_{g\text{-}s}$ = Number of WAP Request packets sent from WAP Gateway to WAP Server

$$T_{wired\text{-}Request} = \frac{([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] + t_2) \text{ x } k}{(1 - \varepsilon_2)^k} +$$

$$(N_{g\text{-}s} - 1) \text{ x } ([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] + t_2)$$

Now

T_s = Time to decompress ROHC + Time to decompress before TLS Layer + Time to serve the request

= $T_{dcompROHC\text{-}s} + T_{decomp\text{-}s} + \text{Constant}_s$ --- (6)

Putting values in equation “1” we get,

$$R_{c\text{-}s} = \text{Constant}_c + T_{comp\text{-}c} + T_{ROHC\text{-}c} + \frac{N_{c\text{-}g}\, t1}{1 - \varepsilon_1} + (N_{g\text{-}s} - 1) \text{ x } ([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} +$$

$$T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] + t_2) + \frac{([(0.0213 \text{x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] + t_2) \text{ x } k}{(1 - \varepsilon_2)^k} +$$

$T_{dcompROHC\text{-}s} + T_{decomp\text{-}s} + \text{Constant}_s$ -- (7)

<u>Server Reply time can be derived as follows:</u>

Time Taken by WAP Reply to go from WAP Server to WAP Client (Re_{s-c}) =

$$T_s + T_{wired-Reply} + T_{wireless-Reply} + T_c \quad \text{----------------------------------} \quad (8)$$

$T_s = Constant_{Reply-s} + T_{comp-s} + T_{ROHC-s}$

$T_{wired-Reply}$ for WAP Reply = time spent over "k" HOPs + time taken by WAP request to go from WAP Server to WAP Gateway excluding the time spent at "k" HOPs.

$T_{wired-Reply}$ for WAP Reply = $T_{HOP} + T_{s-g}$

Where,

$T_{HOP} = [(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC-h} + T_{ROHC-h}] \text{ x } k$

The equation below is derived by using the logic used to derive eq. for $R_{c-s.}$ T_{s-g} for WAP Reply excluding time spent at "k" HOPS is given by

$T_{s-g} = \quad kt_2$

$T_{wired-Reply} = [(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC-h} + T_{ROHC-h}] \text{ x } k + \quad kt_2$

$T_{wired-Reply} = ([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC-h} + T_{ROHC-h}] + t_2) \text{ x } k$

Because of bit error rate "ε_2" of wired network

$$T_{wired-Reply} = \frac{([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC-h} + T_{ROHC-h}] + t_2) \text{ x } k}{(1 - \varepsilon_2)^k}$$

Now for "N_{s-g}" number of requests where

N_{s-g} = Number of WAP Reply packets sent from WAP Server to WAP Gateway

$$T_{wired-Reply} = \frac{([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC-h} + T_{ROHC-h}] + t_2) \text{ x } k}{(1 - \varepsilon_2)^k} +$$

$(N_{s-g} - 1) \text{ x } ([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC-h} + T_{ROHC-h}] + t_2)$

Now,

Time to deliver all the request packets over wireless network = $\frac{N_{g\text{-}c}\ t_1}{1-\varepsilon_1}$

Where $N_{g\text{-}c}$ = Number of WAP Reply packets sent from WAP Gateway to WAP Client.

$Re_{s\text{-}c} = Constant_{Reply\text{-}s} + T_{comp\text{-}s} + T_{ROHC\text{-}s} + (N_{s\text{-}g} - 1) \text{ x } ([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} +$

$T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] + t_2) + \frac{([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] + t_2) \text{ x } k}{(1-\varepsilon_2)^k}$

$+ \frac{N_{g\text{-}c}\ t_1}{1-\varepsilon_1} + T_{dcompROHC\text{-}c} + T_{decomp\text{-}c} + Constant_{Reply\text{-}c}$ -- (9)

Total Response Time for Proposed Architecture = $R_{c\text{-}s} + Re_{s\text{-}c}$ ---------------------------- (10)

Original Architecture Response Time:

Since in original architecture there is no compression/decompression at the WAP Client/WAP Server. So, for the original architecture, equations 7 and 9 can be written as follows:

Time for WAP Request to go from WAP Client to WAP Server ($R_{c\text{-}s\text{-}orig}$) =

$Constant_c + T_{comp\text{-}c} + T_{ROHC\text{-}c} + \frac{N_{c\text{-}g}\ t1}{1-\varepsilon_1} + T_{dcompROHC\text{-}g} + T_{decomp\text{-}g} + (N_{g\text{-}s} - 1) \text{ x}$

$([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + t_2) + \frac{([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + t_2) \text{ x } k}{(1-\varepsilon_2)^k} + Constant_s$

-------------------------------------- (11)

Time for WAP Reply to go from WAP Server to WAP Client ($Re_{s\text{-}c\text{-}Orig.}$) =

$Constant_{Reply\text{-}s} + (N_{s\text{-}g} - 1) \text{ x } ([(0.0213\text{x } L+25) \text{ x } 10^{-6} + t_2)$

$+ \frac{([(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + t_2) \text{ x } k}{(1-\varepsilon_2)^k} + T_{ROHC\text{-}g} + T_{comp\text{-}g}$

$$+\frac{N_{g\text{-}c}\ t_1}{1-\varepsilon_1} + T_{dcompROHC\text{-}c} + T_{decomp\text{-}c} + \ Constant_{Reply\text{-}c}$$

--------------------------------------- (12)

Total Response time for Original Architecture= $R_{c\text{-}s\text{-}Orig.} + Re_{s\text{-}c\text{-}Orig.}$ --------------------- (13)

- Claim "2" can be shown to provide better results than the original architecture by implementing the equations derived below:

First we determine the total time delay for a request to reach the server and back with no gateway between the WAP Client and the WAP server. Since there is no cache at the WAP Gateway, the WAP Response time will be the time to get pages from the server for every WAP Client Request. Time delay for a request can be written as:

Time Taken by WAP Request to go from WAP Client to WAP Server without WAP Gateway ($R_{w/o\ gateway\ c\text{-}s}$) = $T_c + T_{wireless} + T_{wired} + T_s$ -------------------------------------- (14)

Where, T_{wired} = Time spent at "k" HOPs + Time to send WAP request over the wire excluding the time spent at each HOP.

$= T_{HOP} + T_{w/o\ g\text{-}s}$

Where from eq. 4,

$$T_{HOP} = [(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] \text{ x } k$$

Now using the logic used to derive equation 5 above. Since there is no WAP Gateway between the WAP Client and WAP Server:

So, $T_{wireless} + T_{w/o\ g\text{-}s} = t_1 + kt_2$

So we can write,

$$T_{wireless} + T_{w/o\ g\text{-}s} + T_{HOP} = t_1 + kt_2 + [(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] \text{ x } k$$

Because of bit error rate "ε_1" and "ε_2" of wireless and wired network respectively

$$T_{wireless} + T_{w/o\ g\text{-}s} + T_{HOP} = \frac{t_1 + kt_2 + [(0.0213 \text{ x } L + 25) \text{ x } 10^{-6} + T_{dcompROHC\ \text{-}h} + T_{ROHC\ \text{-}h}] \text{ x } k}{(1 - \varepsilon_1)(1 - \varepsilon_2)^k}$$

So, eq. 14 can be written as:

$R_{w/o\ gateway\ c\text{-}s} =$

$$\text{Constant}_c + T_{comp\text{-}c} + T_{ROHC\text{-}c} + \frac{t_1 + kt_2 + [(0.0213 \times L + 25) \times 10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] \times k}{(1-\varepsilon_1)(1-\varepsilon_2)^k} +$$

$T_{dcompROHC\text{-}s} + T_{decomp\text{-}s} + \text{Constant}_s$

Now for $N_{c\text{-}g}$ and $N_{g\text{-}s}$ number of WAP requests packets over the wireless and wired networks:

$R_{w/o\ gateway\ c\text{-}s} =$

$\text{Constant}_c + T_{comp\text{-}c} + T_{ROHC\text{-}c} + (N_{c\text{-}g} - 1) \times t_1 + (N_{g\text{-}s} - 1) \times (t_2 + [(0.0213 \times L + 25) \times 10^{-6}$

$$+ T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}]\ ) + \frac{t_1 + kt_2 + [(0.0213 \times L + 25) \times 10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] \times k}{(1-\varepsilon_1)(1-\varepsilon_2)^k} +$$

$T_{dcompROHC\text{-}s} + T_{decomp\text{-}s} + \text{Constant}_s$ -- (14a)

Time Taken by WAP Reply to go from WAP Server to WAP Client without WAPGateway ($Re_{w/o\ gateway\ s\text{-}c}$) = $T_s + T_{wired} + T_{wireless} + T_c$ -------------------------------- (15)

Using the same logic used to derive eq. 9 we can write:

$Re_{w/o\ gateway\ s\text{-}c} =$

$\text{Constant}_{Reply\text{-}s} + T_{comp\text{-}s} + T_{ROHC\text{-}s} + (N_{s\text{-}g} - 1) \times t_1 + (N_{g\text{-}c} - 1) \times (t_2 + [(0.0213 \times L + 25) \times$

$$10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}]\ ) + \frac{t_1 + kt_2 + [(0.0213 \times L + 25) \times 10^{-6} + T_{dcompROHC\text{-}h} + T_{ROHC\text{-}h}] \times k}{(1-\varepsilon_1)(1-\varepsilon_2)^k}$$

$+ T_{dcompROHC\text{-}c} + T_{decomp\text{-}c} + \text{Constant}_{Reply\text{-}c}$ -- (16)

Total Response Time for Proposed Architecture w/o WAP Gateway ($T_{Resp.\ w/o\ gateway}$) =

$R_{w/o\ gateway\ c\text{-}s} + Re_{w/o\ gateway\ s\text{-}c}$---------------------- (17)

Now if "P" is the probability of cache hit, then for "M" numbers of requests the WAP Response time is as follows:

<u>WAP Gateway without cache:</u>

As, each Request will fetch the Reply message from all the way from the server. So, for "M" WAP Requests:

WAP Response time without cache at WAP Gateway ($D_{\text{w/o cache}}$) = M x $T_{\text{Resp. w/o gateway}}$

$$\text{Average Response Time} = \frac{M \times T_{\text{Resp. w/o gateway}}}{M}$$

Average Response Time = $T_{\text{Resp. w/o gateway}}$ --- (18)

<u>WAP Gateway with cache:</u>

WAP Response time with Cache at the Gateway ($D_{\text{with cache}}$) =

[M x P x (Time for WAP Request to reach the WAP Gateway + Time to construct Reply at WAP Gateway + Time for WAP Response to reach WAP Client + Time to Decompress WAP Reply)] + [M (1-P) x Response Time Without WAP Gateway]

= [M x P x (T_c + T_{wireless} + T_g+ $T_{\text{wireless-Reply}}$ + $T_{\text{Decomp-c}}$ + $T_{\text{DecompROHC-c}}$ + $\text{Constant}_{\text{Reply-c}}$)] + [M (1-P) x ($T_{\text{Resp. w/o gateway}}$)] --- (18a)

Where for "N" number of packets

$$T_{\text{wireless-Reply}} = \frac{N_{\text{g-c}}\ t_1}{1-\varepsilon_1}$$

$$T_{\text{wireless}} = \frac{N_{\text{c-g}}\ t1}{1-\varepsilon_1}$$

So,

$$T_{\text{wireless}} + T_{\text{wireless-Reply}} = (N_{\text{c-g}} + N_{\text{g-c}}) \times \frac{t_1}{1-\varepsilon_1}$$

Also,

$$T_g = T_{\text{dcompROHC-g}} + T_{\text{decomp-g}} + \text{Constant}_g + \text{Constant}_{\text{Reply-g}} + T_{\text{comp-g}} + T_{\text{ROHC-g}}$$

And,

$T_c =$ $Constant_c + T_{comp-c} + T_{ROHC-c}$

Putting above values in eq.18a we get:

$D_{with\ cache} = [M \times P\ (Constant_c + T_{comp-c} + T_{ROHC-c} + (N_{c-g} + N_{g-c}) \times \frac{t_1}{1-\varepsilon_1} + T_{dcompROHC-g} +$

$T_{decomp-g} + Constant_g + Constant_{Reply-g} + T_{comp-g} + T_{ROHC-g} + T_{decomp-c} + T_{decompROHC-c} +$

$Constant_{Reply-c})] + [M\ (1-P) \times (T_{Resp.})]$ --- (19)

So the Average Time Delay = $\frac{D_{with\ cache}}{M}$ --- (20)

Above equation can be divided into request time and response time as follows:

$R_{with-cache} = M \times P\ (T_c + T_{wireless} + T_{decomp-ROHC-g} + T_{decomp-g} + constant_g) + (M \times (1-P)) \times R_{c-s}$

$Re_{with-cache} = M \times P\ (T_{ROHC-g} + T_{comp-g} + constant_{Reply-g} + T_{wireless-Reply} + T_{Reply-c}) + (M \times (1-$

$P)) \times Re_{s-c}$

Adding $R_{with-cache},$ $Re_{with-cache}$ would us eq.19.

- Claim “3” and “5” can be shown to provide better results than the original architecture by using the implementation of equations 10 and 13 derived above.

Total Response Time for Proposed Architecture = $R_{c\text{-}s} + Re_{s\text{-}c}$ ---------------------------- (10)

Total Response time for Original Architecture= $R_{c\text{-}s\text{-}Orig.} + Re_{s\text{-}c\text{-}Orig.}$ --------------------- (13)

CHAPTER 5

RESULTS AND DISCUSSIONS

The values that are considered constant throughout our implementation are as follows:

- The robust header compression (ROHC) and decompression time remain the same for all files.
- The application of ROHC compression on TCP/IP header compresses the header to 1 byte. [23]
- For wired transfer ATM is being used.

  Cell size of ATM is = 5 + 48 = 53 bytes

  Where 48 bytes is the size of the payload and 5 bytes is the ATM header.
- For wired transfer SONET is considered. The data rate of SONET is 155 Mbps.
- For wireless transfer the frame that is being used has the same size as that of the ATM cell.
- For wireless transfer CDMA2000 is considered. The data transfer rate of CDMA2000 is 153 Kbps.
- The transfer time for one ATM cell over the wired network (t_2)

  $$= \frac{58 \times 8}{155E6} = 2.73 \text{ E-6 Sec.}$$
- The transfer time for one frame over the wireless network (t_2)

$$= \frac{53 \times 8}{153E3} = 2.77 \text{ E-3 Sec.}$$

- The time to construct the request and reply at the WAP Client, WAP Gateway and WAP Server = 0.

As the time to construct the request and reply will be same for both, proposed and original architectures. So leaving them out will not affect our calculations. Also in our calculations we are only calculating the WAP Reply time. This is because the WAP request size is very small and will usually fit in one frame or cell. Thus it will not effect the out come of our results. The reply message consists of large amount of data and thus requires a large amount of time to get transferred as compared to the WAP Request.

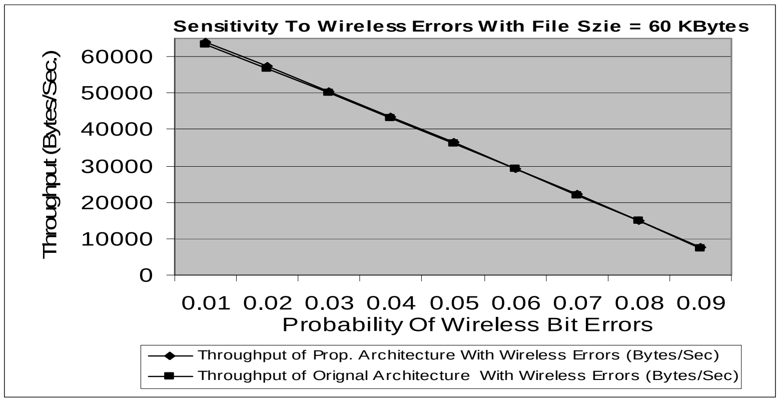
5.1 Effect of Wireless and Wired Errors

Graph 1 shows the throughput of both proposed and original architectures under various wireless bit error rates.

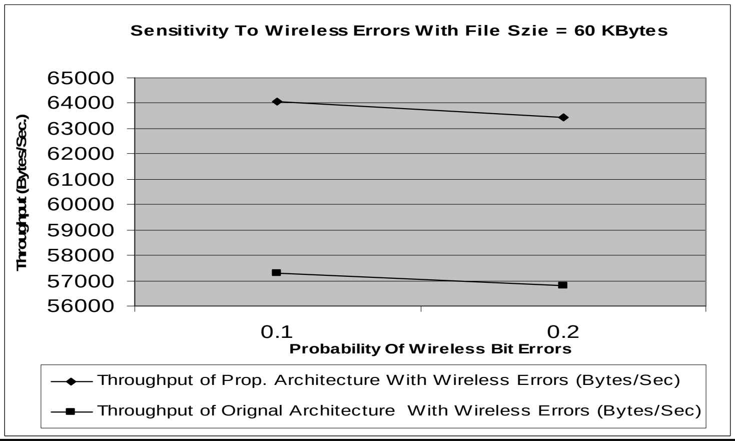
Here, Value of the wired bit error rate is considered constant = 0.01

Graph 1 shows that due to errors in the wireless environment the throughput for both proposed and original architectures decreases sharply. However, under these conditions our proposed architecture still performs better than the original architecture, as can be seen from the graph 1.1 and 1.2 which are the expanded view of graph 1. There is at least a difference of 618 bytes per sec. in the throughput of proposed and original architectures at a wireless bit error rate of 0.1.

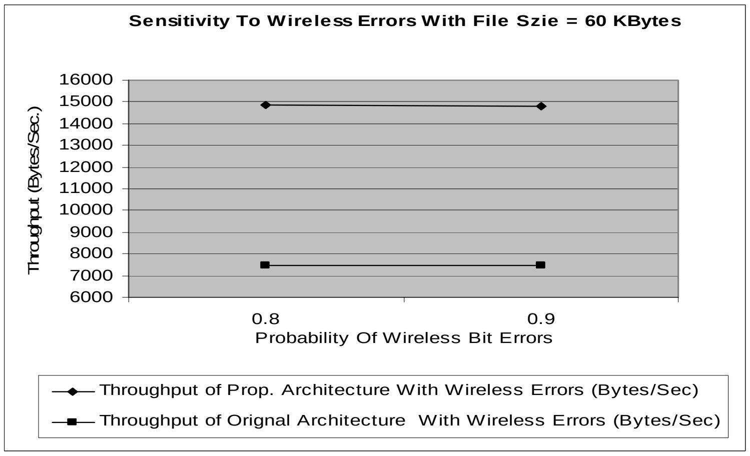
The reason for the sharp decline in the throughput is the lack of pipelining in the wireless environment. In case of wireless network errors the data is transmitted sequentially between the WAP Gateway and the WAP Client. In our architecture there is pipelining from WAP Client till WAP Gateway but in case of wireless errors the pipeline will stall as it has to send requested packets that could not make it to the WAP Client.



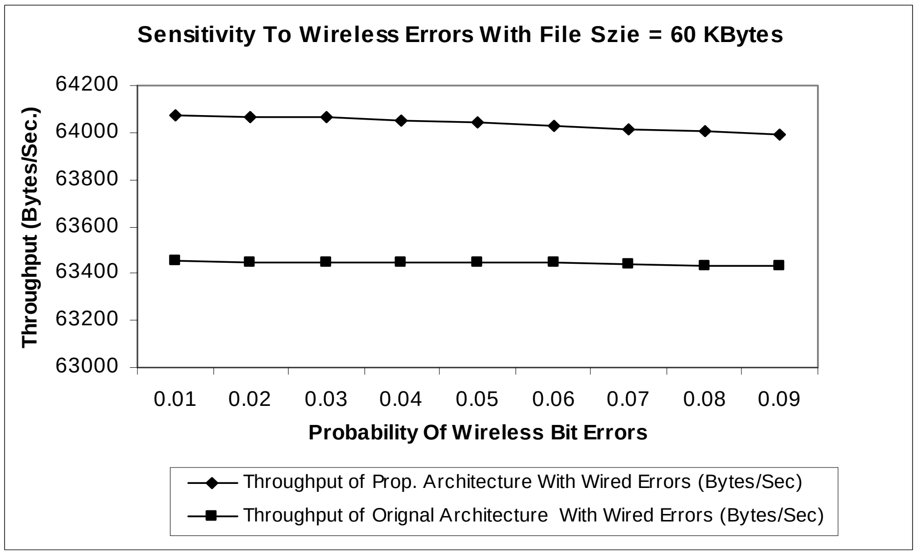
Graph 1: Comparison of Proposed and Original architectures with wireless network errors.



Graph 1.1 Expanded View of Graph 1 for Bit Error Rate of 0.1 and 0.2



Graph 1.2 Expanded View of Graph 1 for Bit Error Rate of 0.8 and 0.9



Graph 2: Comparison of Proposed and Original architectures with wired network errors.

Graph 2 shows gain in the throughput using our proposed architecture. This graph shows the throughput of both proposed and original architectures under various wired bit error rates.

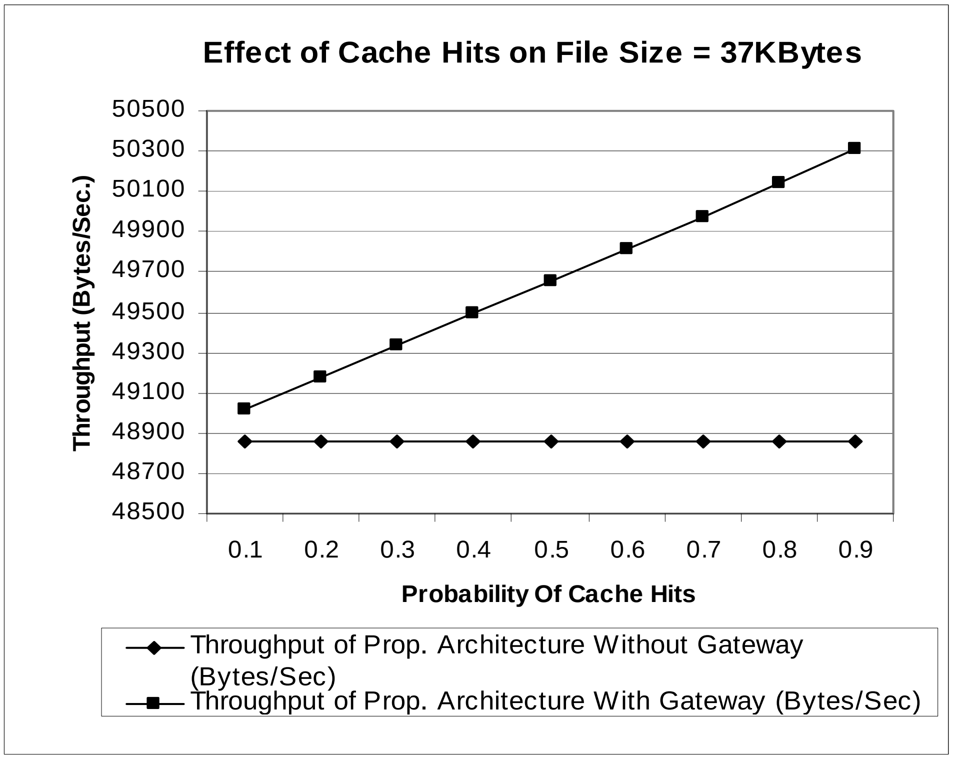
Here, Value of the wireless bit error rate is considered constant = 0.1

The reason for this gain is the compression of data and HTTP headers at the WAP Server and the existence of pipelining from WAP Server till the WAP Client. Due to compression there are fewer data packets to be transferred and due to pipelining it takes less time for packets to be delivered from WAP Server to WAP Client. Whereas, in the original architecture pipelining exists only in the wired network. All the data is sent uncompressed to the WAP Gateway. All the data is then collected at the WAP Gateway for compression and then is sent out to the WAP Client.

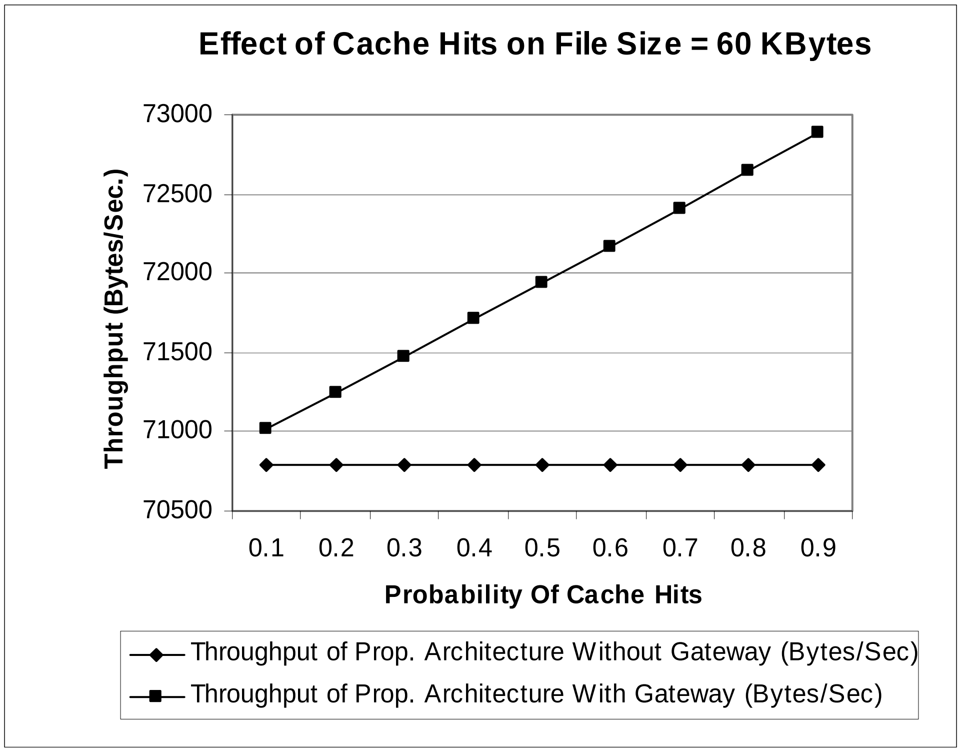
Now in case of errors in the wired network there is less effect on the throughput of both architectures. The reason for this is the existence of pipelining and the high transfer speed of wired network.

5.2 Effect of Cache Hits

Graphs below show the effect of different cache hit probabilities. These graphs indicated the significance of having cache at the gateway. As can be seen from the results, an increase in cache hit results in an increase of throughput. The reason for this improvement is that if we do not have a cache at the WAP Gateway then every request message has to go all the way to the server to get a reply. But the existence of a cache at the WAP Gateway means that some of the requests will get their reply from the WAP Gateway. This results in improved response time.



Graph 3: Comparison of Proposed architecture with and without WAP Gateway for different Cache hit Probabilities for file size = 37 KBytes.



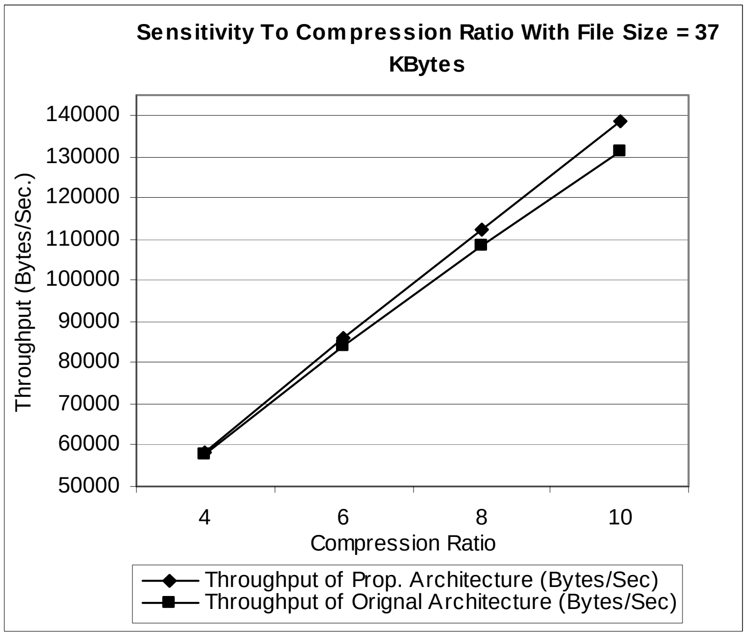
Graph 4: Comparison of Proposed architecture with and without WAP Gateway for different Cache hit Probabilities for file size = 37 KBytes.

So, it is good to have a cache at the WAP Gateway with some cache policy to store pages.

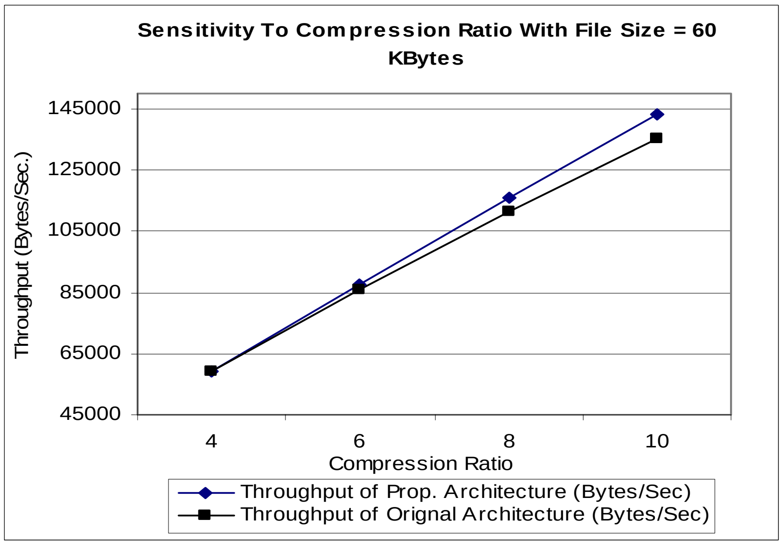
Data packets are stored in decompressed format in the cache. So for a WAP Client request, packets are compressed and then sent to the WAP Client over the wireless network. If data packets are stored in the compressed format then WAP Client request will not be able to identify those packets as the data requested by the client.

The presence of a cache means more work for the WAP Gateway. It will require more memory to store data. Also, it will have to compress the data for any cache hit to send it to the WAP Client. This overhead is not difficult to overcome due to the availability of hardware that is not only cheap to buy but is getting cheaper every day. On the other hand bandwidth is a very precious and expensive resource and must be used in an efficient way.

5.3 Effect of Compression Ratios



Graph 5: Comparison of Proposed and Original architecture for various compression ratios for data of size = 37 KBytes.



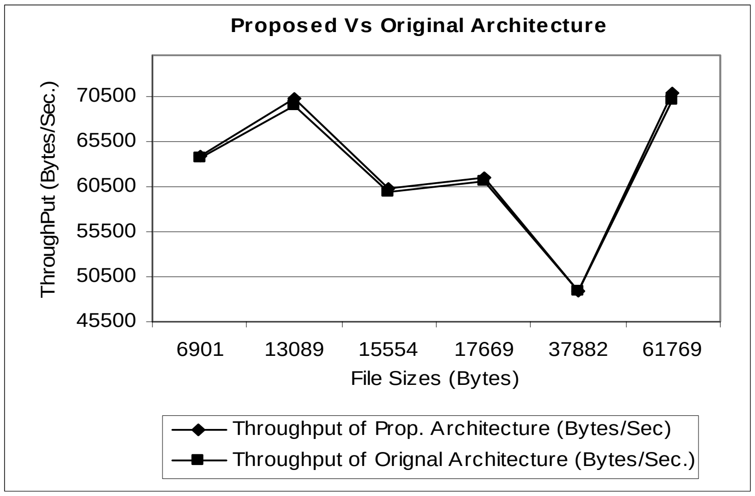
Graph 6: Comparison of Proposed and Original architecture for various compression ratios for data of size = 60 KBytes.

Graphs above are for various data compression ratios. Our purpose here is to see the effect of compression on throughput. The compressed file sizes considered in graphs are theoretical file sizes and are not generated by any particular compression algorithm. The time to compress/decompress the files is also considered to be the constant. So for calculating the affects of compression the only varying quantity is the file size under different compression ratios.

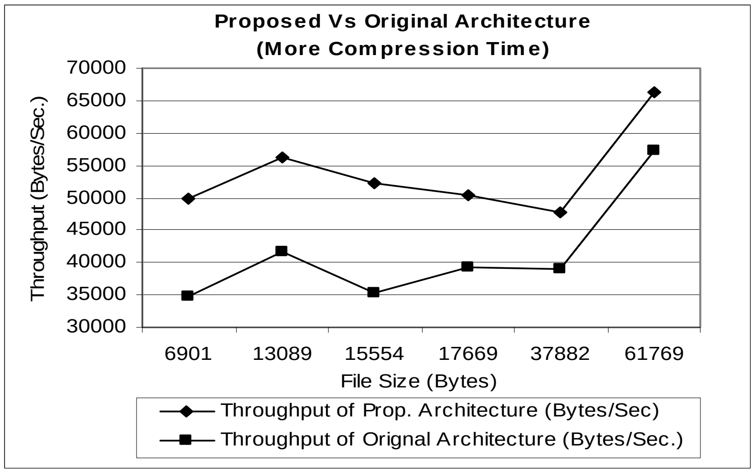
Graphs 5, 6 above show that the smaller the compressed file size the better is the throughput of the proposed architecture. So using an efficient lossless compression algorithm is an important factor in the proposed architecture. Also we can observe from the graphs that our proposed architecture perform better for files that have compression ratios above 4.

5.4 Effect of Compression Time

Another factor that will affect both the proposed and original architectures is the compression time needed to compress data and HTTP headers. This compression is applied after the TLS layer. An increase in compression time will reduce the overall throughput for both architectures under consideration. But as shown in the graphs below the throughput of our proposed architecture is a lot better than the throughput of the original architecture. So this shows that the proposed architecture tends to perform better than the original architecture under increased data and HTTP header compression time.
Here in our analysis we considered the wireless and wired errors to be zero. Also, the compression time need to compress data and HTTP headers is ten times more for graph 8 than the original experimental time used to draw graph 7.



Graph 7: Proposed Vs. Original Architecture with less time required to compress data and HTTP headers.



Graph 8: Proposed Vs. Original Architecture with more time required to compress data and HTTP headers.

The reason for the gain in our proposed architecture is the existence of pipelining all the way from the WAP Server to the WAP Client. But in the original architecture all the packets have to be collected at the WAP Gateway for compression before sending them out to the WAP Client. So the increase in compression in the original architecture has more effect on throughput as compared to our proposed architecture.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

Our proposed architecture shows improvement over the original architecture [1] under the following conditions.

1. The compression algorithm should at least give a compression ratio of 4 or more to show any significant improvement over the original architecture.
2. The ROHC compression and Decompression should take a very small amount of time to compress and decompress the IP headers.

Graph 5 and 4 show that the throughput is directly proportional to the size of the compressed data. As the size of the file decreases our proposed architecture show more and more improvement where as there is no effect on the original architecture.

Compression and decompression time of the ROHC algorithm will affect our architecture because every data packet has to be decompressed at each HOP for destination lookup and other functionalities and then compressed again. If this compression and decompression time is significantly large then the efficiency of our proposed architecture will decrease.

The increase in compression time of data and HTTP header tends to reduce the overall throughput but the performance of our proposed architecture is better than the original architecture.

So, we can say that the overall performance of our architecture is better than the original architecture. We have used only the reply time to calculate our results. The WAP request is of small size and generally consists of 1 to 2 packets. Due to the availability of the high speed wireless and wired networks, WAP Request compression is not necessary. This compression would be beneficial if the low speed networks like IS-95 were being used.

The performance of our architecture can be further improved by having the data in already compressed form at the WAP Gateway. This will help in reducing the transfer time thus will increase the overall throughput of the proposed architecture.

In our implementation data is transferred over the wired and wireless internet in compressed form. So it has to be decompressed at the WAP Client and WAP Server side. This process takes some extra computations. Since power is a precious resource for the WAP Client, someone might argue that compression should not be done as it consumes more power. But, "Bandwidth is the most costly resource in cellular links. Processing power is very cheap in comparison. Implementation or computational simplicity of a header compression scheme is therefore of less importance than its compression ratio and robustness." [22][23].

By using our proposed architecture we save on bandwidth power. This gain in bandwidth power makes the power gain at the WAP Client less significant.

6.1 Drawbacks

Throughout our simulations of results we have considered:

- Time to compress and decompress IP headers to be constant. But in real life scenario this is not true as this time is dependent on factors such as load on the WAP Server, WAP Client and at each HOP.
- The size of the data packet to be equal to the size of the ATM Cell. In real life this is not the case.
- The time to construct request and reply messages at the WAP Client and WAP Server are also considered as zero.
- The file sizes that we used were considered to have the HTTP headers included in them i.e. File Size = HTTP header + Data
- We also used the size of the IP header to be equal to 1. This is because of the unavailability of data. We got this value from a white paper "The concept of robust header compression, ROHC" [23].
- We have only used GZIP to compress the data and HTTP headers. The reasons for using this algorithm is its availability and that it uses Deflate algorithm which is a variation of LZ77 (Lempel-Ziv 1977) and Huffman coding to compress data. According to a survey done by Choi [25], Lempel-Ziv is the best algorithm for text compression [25].

6.2 Future Enhancements

For future enhancements:

1- Use actual values for ROHC compression and decompression and see their effects.

2- Use different types of compression algorithms to compress the data and HTTP headers.

3- Try our proposed architecture on different wired and wireless networks and environments.

4- Study the effects of push functionality on our proposed architecture.

5- Study effects of having data in already compressed from at the WAP Server.

6- Study the effects of increased computation due to compression and decompression on the power consumption of mobile devices.

REFERENCES

[1] Zhanping Yin, Victor C. M. Leung, "A proxy architecture to enhance the performance of WAP 2.0 by data compression", WCNC 2003 - IEEE Wireless Communications and Networking Conference, vol. 4, no. 1, Mar 2003, pp. 1322-1327.

[2] Eetu Ojanen, Jari Veijalainen, "Compressibility of WML and WMLScript byte code: Initial results", Research Issues in Data Engineering, 2000. RIDE 2000 Proceedings. Tenth International Workshop on, 28-29 Feb. 2000 Pages: 55 - 62

[3] Stefan Aust, Nikolaus A. Fikouras, and Carmelita Görg, "Enabling Mobile WAP Gateways using Mobile IP", Proceedings of the European Wireless 2002, Florence, Italy, February 2002. http://docenti.ing.unipi.it/ew2002/proceedings/157.pdf

[4] Yi-Ming Wen, Hsing Mei, "Quick Implementation of a WAP Push Gateway", Proceedings of 2000 Workshop on Internet and Distributed Systems (WINDS), Tainan, Taiwan, May 2000, pp. 486-493.

[5] Vijay Kumar, Srinivas Parimi, and Dharma P. Agrawal, "WAP: Present and Future", Pervasive Computing, IEEE, Volume: 2, Issue: 1, Jan-Mar 2003 pp. 79 - 83.

[6] WAP Forum, "WAP architecture specification", version 12-July-2001, http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html, July 2001

[7] WAP Forum, “WAP 2.0 technical white paper”, 01-Jan-2002, http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[8] WAP Forum, “Wireless profiled TCP”, 31-March-2001, http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[9] WAP Forum, “Wireless Profiled TCP”, 31-March-2001, http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[10] WAP Forum, “WAP Push Architectural Overview” 03-Jul-2001, http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[11] Peter Stuckmann, Christian. Hoymann., “Performance evaluation of WAP-based applications over GPRS”, in Proc. IEEE/ICC 2002, New York, May 2002, pp. 3356-3360.

[12] Ric Howell, “WAP Overview”, http://www.perfectxml.com/Conf/Wrox/Files/howell2text.pdf

[13] NOKIA, "Transition to mobile application over TCP/IP", http://www.nokia.com/nokia/0,,32921,00.html

[14] G. Radhamani, K. Ramasamy, “Security issues in WAP WTLS protocol”, Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference on, Volume: 1, 29 June-1 July 2002 pp. 483-487 vol.1.

[15] Andrew G. Blank , “TCP/IP JumpStart”, SYBEX Inc., 1151 Marina Village ParkWay, Alameda, CA 94501.

[16] Ericsson, “WAP architecture overview”, http://www.ericsson.com/

[17] WAP Forum, “WAP TLS Profile and Tunneling Specification ”, 11-April-2001,

http://www.openmobilealliance.org/tech/affiliates/wap/wap-219-tls-20010411-a.pdf

[18] http://www.gzip.org

[19] Peter Deutsch, ”DEFLATE Compressed Data Format Specification version 1.3”, RFC 1951, May 1996

[20] http://www.kannel.org

[21] “Key User Features of GPRS”, http://www.gsmworld.com/technology/gprs/intro.shtml#1.

[22] Carsten Bormann etc., “Robust header compression (ROHC)”, RFC3095, July 2001.

[23] EFFNET AB, “The concept of robust header compression, ROHC”, Feburary-2004, http://www.effnet.com.

[24] Konstantina Papagiannaki etc., “Analysis of Measured Single-Hop Delay from an Operational Backbone Network”, 2002 IEEE, http://www.ieee-infocom.org/2002/papers/889.pdf

[25] Sunny Choi, “A Comparison of Methods For Text Compression”, December-1989 Master’s Thesis, Computer Science Department, Oklahoma State University.

VITA

Kashif Khan

Candidate for the Degree of

Master of Science

Thesis: EFFECTS OF COMPRESSION ON SYSTEM THROUGHPUT IN WIRELESS APPLICATION PROTOCOL (WAP) 2.0 ARCHITECTURE

Major Field: Computer Science

Biographical:

Personal Data: Born in Lahore, Pakistan, May 1976, the son of Mr. and Mrs. Inayat Ullah Khan.

Education: Graduated from Cathedral High School, Lahore, Pakistan, in 1990. Received Bachelor of Science Degree in Math and Physics from Government College Lahore, Pakistan in 1994. Received Bs. Mechanical Engineering from University of Engineering and Technology Taxila, Pakistan, in 1999. Completed the requirements for Master of Science at Oklahoma State University in May, 2005

.