

Implementation of WAP Gateway Technologies Through Wireless Communication

A. K.Muruganandam¹, B. V.Palanisamy², C. A.Krishnan³ and D.R.Rajesh⁴

¹ VIT University / ECE, Vellore, India

Email: kmuruganandam@vit.ac.in

⁴ VIT University / ECE, Vellore, India

Email: rajesh@vit.ac.in

Abstract- The Wireless Application Protocol (WAP) is an emerging standard for the deployment of data oriented applications in wireless environments. Although some components of the WAP suite have been developed, it lacks a complete general architecture integrating all the protocols, software components of both the Internet and wireless contexts in a transparent way. The protocol stack consists of miniaturized versions of UDP, TCP, SLP and HTTP protocols with reduced header sizes and complexity to make the protocols usable in wireless sensor networks. This paper presents a general architectural framework to develop and deploy portable applications and services accessible by WAP compliant mobile terminals, extending end-to-end services between terminal and business applications.

Keywords— WAP, vendor development protocols, Nano UDP, TCP, and SLP.

I. INTRODUCTION

There are more than 300 million cell phone users in the world. There are more also millions of Internet users in the world. These two markets are rapidly converging on the same spot: small, lightweight, inexpensive mobile computing devices that are equally suitable for high-quality voice communication, modest bandwidth data communication, seamless Internet connectivity, and access to the Internet services such as e-mail and content. Few years ago the some industry leader company predict this convergence and started a collaborating to design architecture to delivery advanced data services to cell phones. This architecture, which draws heavily from existing Internet technologies, is based on a protocol stack called the Wireless Application Protocol (WAP) . WAP sits on top of a variety of wireless carriers. Today the most widespread is the GSM infrastructure. It provides services such as compression, encryption, the integration of telephony and data services and the Wireless Application Architecture (WAE) for building applications.

K. Muruganadam Research Scholar Anna University Chennai. (Phone: 9443441729 email: kmuruganandam@vit.ac.in).

R.Rajesh Research Scholar VIT University, Vellore (Phone: 9894428881 email: rajesh@vit.ac.in)

II. EXISTING ARCHITECTURE AND PROBLEM

To deploy applications for wireless terminals it is not sufficient to simply guarantee access to data (e.g. web or database contents), but it is necessary to take into account the constraints of wireless connections, both in architectures and applications. The main constraints of wireless networks are the following: Low bandwidth Current values range from 0,3 to 10 Kbits/s, some order of magnitude lower than wired ones. High latency. Latency depends by the type of medium being crossed, and by the cell load and unsteady connection. Main causes are the blank out period during handover, the limited duration of the battery and exiting from the covered area. High Transmission Bit Error Rate (BER). Error rates are greater than in wired networks (typical phenomena are fading and Interference) and it is more difficult to guarantee Quality of Service and low predictable service availability. Wireless networks can suffer short or long periods of inaccessibility, due to congestions or faults. The main constraints of wireless terminals are: Small, low resolution displays and limited user-input facilities. Displays have few lines of text and low resolution. Terminals with lack of mouse and keyboard are thus requiring new User Interface different by typical GUI. Current U0 devices are touch-screen and voice menu have Limited computational resources (CPU, RAM), duration and power of batteries and many kinds of terminals. Wireless terminals are available in different models (handheld, laptop, communicator, smart phone) and architectures, which can cause interoperability problems.

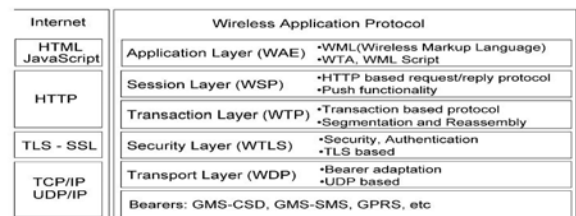


Fig. 1 Layered Architecture for WAP

III.BASIC COMPONENTS OF THE WAP FRAMEWORK

A micro-browser (WML user-agent, see below) hosted by the mobile terminal (client-side); a WAP gateway allowing the inter working operation between the

wireless network and others kinds of data networks (e.g. TCP/IP networks); a layered stack of (WAP) protocols specifically designed for wireless environments. Moreover, a highly interoperable application environment (Wireless Application Environment, WAE) is provided. WAE defines an XML-compliant markup language named WML (Wireless Mark-up Language), and the associated scripting language WML Script. The p-browser is a simplified web browser executed on a terminal able to display WML contents and to execute WML Script code, which communicates with the WAP gateway using the Wireless Application Protocol, on different wireless transports. WML is a simplified data description language able to face the constraints of wireless environments, which prevent the effective support of HTML. Moreover, WML defines a new WAP-specific User Interface (UI) model based on the Card & Deck metaphor. A WML document is a deck composed by a set of cards. A card is a unit of interaction between user and application (e.g. a menu, a form to input data or simply text), whereas a deck is a set of logically related cards and represents the smallest information sent by the application to the terminal. Decks and cards are referred to by URLs. Likewise HTML, anchors within an URL allow references to specific cards within the current deck.

Basic characteristics of WML are: inter-card browsing; chronological management of decks; formatting and layout; management of events; WML variables are declared locally to the terminal and are engaged by the p-browser. Variables and their runtime values are stored in a terminal memory location named browser context. These values can be accessed in any point of the WMLNML Script code. The main advantage in having local variables is the possibility to realize efficient mechanisms to manage and maintain the state of application. In fact, using state variables limits the transmission of session variables between server and client. The WAP gateway carries out inter working operations between GSM and TCP/IP (e.g. Internet) networks:

0 protocol conversion between the WAP layers (WSP, WTP, WTLS, and WDP) and the WWW layers (HTTP + TCP/IP).

0 format conversion - i.e. the on-the-fly, dynamic translation of contents between HTML and WML (HTML transcoder module). support of text (Unicode) and images; support of local variables and timers. The communication between gateway and mobile terminal occurs through the following software modules:

0 Encoder - that encodes data using a particular compression technique to minimize the number and dimension of packets transmitted via the radio Interface.

0 Decoder - that carries out the inverse decoding operation.

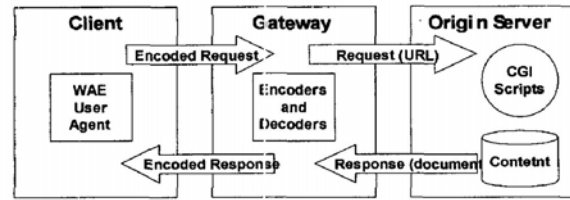


Fig 2. Gateway between client and server

IV. ARCHITECTURE AND IMPLEMENTATION OF TRANSPORT PROTOCOLS.

Our transport protocols consist of the reliable byte-stream based nanoTCP and the best effort nanoUDP. In this section we present the major architectural differences between nanoTCP, nanoUDP and the original IP transport protocols they are based on. For the ease of comprehension, we use the standard TCP/IP terms segment for the formatted block of data carried by nanoTCP and datagram for the one carried by nanoUDP. We have implemented our protocols in nesC, a component based enhancement of ANSI-C for embedded environments, in the TinyOS-2.x [13] framework.



Fig 3. NAM output for WAP Gateway

We used the 4th generation Telos motes using the default MAC protocol, Berkley's MAC (B-MAC) and the default setting of disabled ARQ. Telos motes are equipped with the MSP430 microcontroller having 10 KB of RAM and 48 KB of ROM. In Table I we show the memory footprints of our transport protocols; notice the low memory consumption for both codes in RAM and ROM. The figure below depicts more about the WAP gateway, wherein the vendor development protocols are integrated in subnets of the available LAN.

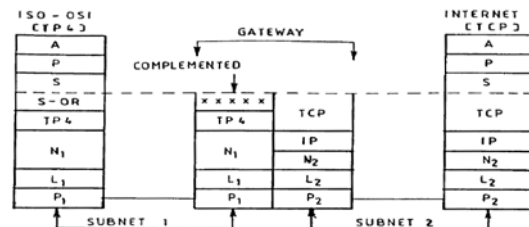


Fig 4. Gateway connected with protocols

A. NANO TCP

The nanoTCP is a simplified version of the reliable byte stream TCP protocol [14]. Similarly to TCP,

the segment structure in Figure 1 starts with the flag field which provides information about the type of the packet. Each segment contains an 8-bit source and destination port number to identify the sending and receiving applications. These two values along with the source and destination addresses provided by the lower layers of the stack uniquely identify a connection. The nanoTCP reliability resides in its retransmission capability provided by the sequence and acknowledgment numbers. Congestion control is not considered, as in WSNs losses are very likely due to a bad link quality and this phenomenon invokes erroneous congestion response. The nanoTCP has three ways- handshake connection, establishment and connection termination work in TCP. While in TCP the flow of data is controlled by an adjustable window size, in nanoTCP we apply a fixed window size again for simplicity. The nanoTCP implementation consists of the module NanoTCPC and definitions in NanoTCPC providing an interface towards the application process through a set of calls and events. Where the nanoTCP communicates with an application called NanoApp 1 and vice versa using port number 22. When the application has data to send NanoApp 1 calls the command open-cx asking the nanoTCP to start the establishment of the connection. Upon receiving this call the three-way handshake takes place. The event cx-ready is signaled informing the application that the connection is open.

The NanoApp 1 passes the data to the nanoTCP which will start sending it in segments and waiting for their respective acknowledgments. Directly after the last acknowledgment is received, the transport signals a send-done event to NanoApp 1 informing it that all the data have been successfully sent. Upon receiving this event, the NanoApp 1 calls the command close-cx for the nanoTCP to close this direction of the connection. As a reaction on this call the stack starts the connection termination procedure and signals cx-closed back to the NanoApp 1 after the connection is closed.

B. NANO UDP

The nanoUDP is a simplified version of the unreliable datagram oriented UDP protocol. The datagram format of nanoUDP we point out especially the small

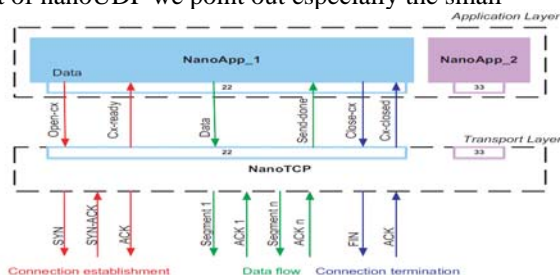


Fig 5. Architecture of Nano UDP

header size of this protocol, which is less than half the size of the nanoTCP header. The five octet header begins with a one byte flag field used for the fragmentation process followed by the length and port fields as in nanoTCP. The nanoUDP implementation consists of the module

NanoUDPM and definitions in NanoUDPC, which provide a simple interface to the application implementing only one command and one event. The communication that occurs between NanoApp 1 and nanoUDP through the port number 44. Since nanoUDP is a connectionless transport protocol, the management of a connection is omitted. On the sender side NanoApp 1 calls directly the command send-data passing a pointer to the data and the nanoTCP starts fragmenting the data and sending it to the lower layers. In our application we limit the number of fragments to eight. On the receiver side, only when all the fragments are completed, nanoUDP signals receive-data event to the application. In the opposite case when at least one fragment is lost, nanoUDP discards the received data and does not signal the application.

V.CONCLUSION

This paper presented an integrated general architecture to develop and deploy portable applications and services for WAP mobile terminals. The architecture is object-based and follows a modular approach to the design of applications, thus allowing application portability and component reusability. The integration between WAP, WWW and all the supporting protocols allows end-to-end services between terminal and business applications for the development of transactional services. This system if implemented allows fast and flexible prototyping of new services for wireless terminals.

REFERENCES

[1] Wireless Application Protocol Architecture specification - WAP Forum [http:// www. wapforum.org/](http://www.wapforum.org/).
 [2] Wireless Application Environment Specification Forum. <http://www.wapforum.org/what/technical.htm>.
 [3] Wireless Markup Language – WAP Forum. [http:// www. wapforum.org/what/technical.htm](http://www.wapforum.org/what/technical.htm).
 [4] Wireless Markup Language Script – WAP Forum. <http://www.wapforum.org/what/technical.htm>.
 [5] Thomas Enderes, Design and Implementation of a Multiplexing Framework for the Wireless application Protocol, Tech. Rep., University of Karlsruhe-Institute for Communications Engineering, May 27, 1999.
 [6] Kyriacos Sabatakakis, Marcel Zumbuhl, Steffen Krotsch, Mobile ecommerce, Andersen Consulting, Zurich, Switzerland.