M.Sc Computer Science-IST YEAR

Mobile Communication

Unit V

Wireless Application Protocol: Wireless Application Protocol(WAP)-Architecture-XML-WML Script-Applications

10.3 Wireless application protocol (version 1.x)

Wireless Application Protocol (WAP) is a technical standard for accessing information over a mobile wireless network. A WAP browser is a web browser for mobile devices such as mobile phones that uses the protocol. Introduced in 1999,[1] WAP achieved some popularity in the early 2000s, but by the 2010s it had been largely superseded by more modern standards. Most modern handset internet browsers now fully support HTML, so they do not need to use WAP markup for web page compatibility, and therefore, most are no longer able to render and display pages written in WML, WAP's markup language.[2]

10.3.1 Architecture

Layers of WAP Protocol

Application Layer

Wireless Application Environment (WAE). This layer is of most interest to content developers because it contains among other things, device specifications, and the content development programming languages, WML, and WMLScript.

Session Layer

Wireless Session Protocol (WSP). Unlike HTTP, WSP has been designed by the WAP Forum to provide fast connection suspension and reconnection.

Transaction Layer

Wireless Transaction Protocol (WTP). The WTP runs on top of a datagram service, such as User Datagram Protocol (UDP) and is part of the standard suite of TCP/IP protocols used to provide a simplified protocol suitable for low bandwidth wireless stations.

Security Layer

Wireless Transport Layer Security (WTLS). WTLS incorporates security features that are based upon the established Transport Layer Security (TLS) protocol standard. It includes data integrity checks, privacy, service denial, and authentication services.

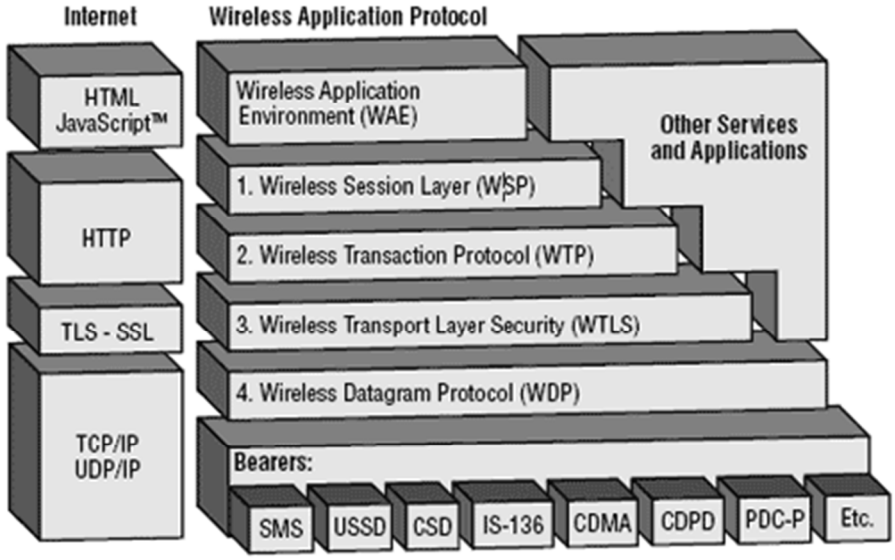
Transport Layer

Wireless Datagram Protocol (WDP). The WDP allows WAP to be bearer-independent by adapting the transport layer of the underlying bearer. The WDP presents a consistent data format to the higher layers

of the WAP protocol stack, thereby offering the advantage of bearer independence to application developers.

Each of these layers provides a well-defined interface to the layer above it. This means that the internal workings of any layer are transparent or invisible to the layers above it. The layered architecture allows other applications and services to utilise the features provided by the WAP-stack as well. This makes it possible to use the WAP-stack for services and applications that currently are not specified by WAP.

The WAP protocol architecture is shown below alongside a typical Internet Protocol stack.



Note that the mobile network bearers in the lower part of the figure above are not part of the WAP protocol stack.

Wireless Application Environment (WAE), the uppermost layer in the WAP stack, provides an environment that enables a wide range of applications to be used on the wireless devices. We have earlier discussed about the WAP WAE programming model. In this chapter, we will focus on the various components of WAE.

Components of WAE

Addressing Model

A syntax suitable for naming resources stored on servers. WAP use the same addressing model as the one used on the Internet that is Uniform Resource Locators (URL).

Wireless Markup Language (WML)

A lightweight markup language designed to meet the constraints of a wireless environment with low bandwidth and small handheld devices. The Wireless Markup Language is WAP's analogy to HTML used on the WWW. WML is based on the Extensible Markup Language (XML).

WMLScript

A lightweight scripting language. WMLScript is based on ECMAScript, the same scripting language that JavaScript is based on. It can be used for enhancing services written in WML in the way that it to some extent adds intelligence to the services; for example, procedural logic, loops, conditional expressions, and computational functions.

Wireless Telephony Application (WTA, WTAI)

A framework and programming interface for telephony services. The Wireless Telephony Application (WTA) environment provides a means to create telephony services using WAP.

Hardware and Software Requirement

At minimum developing WAP applications requires a web server and a WAP simulator. Using simulator software while developing a WAP application is convenient as all the required software can be installed on the development PC.

Although, software simulators are good in their own right, no WAP application should go into production without testing it with actual hardware. The following list gives a quick overview of the necessary hardware and software to test and develop WAP applications –

- A web server with connection to the Internet
- A WML to develop WAP application
- A WAP simulator to test WAP application
- A WAP gateway
- A WAP phone for final testing.

Microsoft IIS or Apache on Windows or Linux can be used as the web server and Nokia WAP Toolkit version 2.0 as the WinWAP simulator.

Please have look at WAP - Useful Resources to find out all the above components.

10.3.2 Wireless Datagram Protocol

Wireless Datagram Protocol (**WDP**)[1] defines the movement of information from receiver to the sender and resembles the User Datagram Protocol in the Internet protocol suite.

The Wireless Datagram Protocol (WDP), a protocol in WAP architecture, covers the Transport Layer Protocols in the Internet model.

As a general transport service, WDP offers to the upper layers an invisible interface independent of the underlying network technology used.

In consequence of the interface common to transport protocols, the upper layer protocols of the WAP architecture can operate independently of the underlying wireless network.

By letting only the transport layer deal with physical network-dependent issues, global interoperability can be acquired using mediating gateways.

10.3.3 Wireless Transport Layer Security (WTLS)

Wireless Transport Layer Security (**WTLS**) is a security protocol, part of the Wireless Application Protocol (WAP) stack. It sits between the WTP and WDP layers in the WAP communications stack.

W*ireless* **T***ransport* **L***ayer* **S***ecurity* (**WTLS**) is the security layer of the WAP, providing privacy, data integrity and authentication for WAP services. WTLS, designed specifically for the wireless environment, is needed because the client and the server must be authenticated in order for wireless transactions to remain secure and because the connection needs to be encrypted.

For example, a user making a transaction with a bank over a wireless device needs to know that the connection is secure and private and not subject to a security breach during transfer (sometimes referred to as a *man-in-the-middle attack*). WTLS is needed because mobile networks do not provide complete end-to-end security.

WTLS is based on the widely used TLS v1.0 security layer used in Internet. Because of the nature of wireless transmissions, modifications were made to the TLS v1.0 in order to accommodate for wireless' low bandwidth, datagram connection, limited processing power and memory capacity, and cryptography exporting restrictions.

10.3.4 Wireless transaction protocol

The wireless transaction protocol (WTP) is on top of either WDP or, if security is required, WTLS (WAP Forum, 2000d).

WTP has been designed to run on very thin clients, such as mobile phones. WTP offers several advantages to higher layers, including an improved reliability over datagram services, improved efficiency over connection-oriented services, and support for transaction-oriented services such as web browsing. In this context, a transaction is defined as a request with its response, e.g. for a web page.

WTP offers many features to the higher layers. The basis is formed from three classes of transaction service as explained in the following paragraphs. Class 0 provides unreliable message transfer without any result message. Classes 1 and 2 provide reliable message transfer, class 1 without, class 2 with, exactly one reliable result message (the typical request/response case).

WTP achieves reliability using duplicate removal, retransmission, acknowledgements and unique transaction identifiers. No WTP-class requires any connection set-up or tear-down phase. This avoids unnecessary overhead on the communication link. WTP allows for asynchronous transactions, abort of transactions, concatenation of messages, and can report success or failure of reliable messages (e.g., a server cannot handle the request). To be consistent with the specification, in the following the term initiator is used for a WTP entity initiating a transaction (aka client), and the term responder for the WTP entity responding to a transaction (aka server). The three service primitives offered by WTP are TR-Invoke to initiate a new transaction, TR-Result to send back the result of a previously initiated

transaction, and TR-Abort to abort an existing transaction. The PDUs exchanged between two WTP entities for normal transactions are the invoke PDU, ack PDU, and result PDU.

10.3.4.1 WTP class 0

The WTP entity at the initiator sends an invoke PDU which the responder receives. The WTP entity at the responder then generates a TR-Invoke.ind primitive with the same parameters as on the initiators side, except for which is now the local handle for the transaction on the responders side. In this class, the responder does not acknowledge the message and the initiator does not perform any retransmission. Although this resembles a simple datagram service, it is recommended to use WDP if only a datagram service is required. WTP class 0 augments the transaction service with a simple datagram like service for occasional use by higher layers.

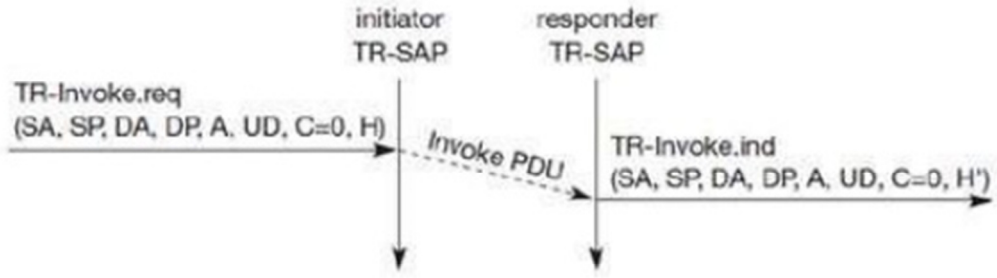


Fig 4.6 Basic Transaction , WTP Class 0

10.3.4.1 WTP class 1

WTP class 1 Class 1 offers a reliable transaction service but without a result message. Again, the initiator sends an invoke PDU after a TR-Invoke.req from a higher layer.

This time, class equals ,,1, and no user acknowledgement has been selected as shown in Figure 10.15. The responder signals the incoming invoke PDU via the TR-Invoke.ind primitive to the higher layer and acknowledges automatically without user intervention. The specification also allows the user on the responders side to acknowledge, but this acknowledgement is not required. For the initiator the transaction ends with the reception of the acknowledgement. The responder keeps the transaction state for some time to be able to retransmit the acknowledgement if it receives the same invoke PDU again indicating a loss of the acknowledgement.

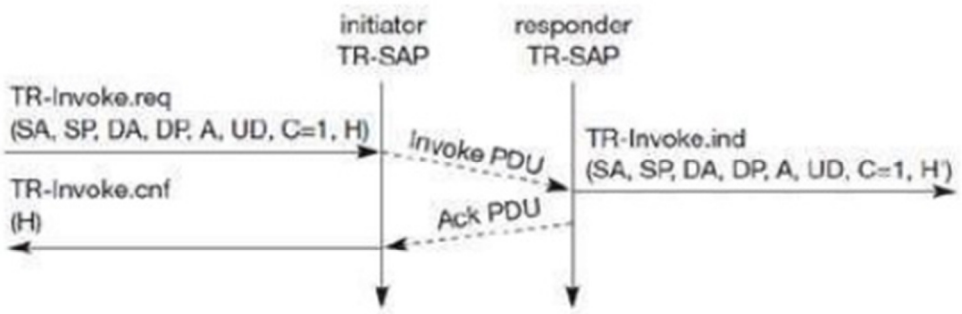


Fig 4.7 Basic Transaction , WTP Class 1, no user Acknowledgement

10.3.5 Wireless Session Protocol

The **wireless session protocol (WSP)** has been designed to operate on top of the datagram service WDP or the transaction service WTP (WAP Forum, 2000e). For both types, security can be inserted using the WTLS security layer if required. WSP provides a shared state between a client and a server to optimize content transfer. HTTP, a protocol WSP tries to replace within the wireless domain, is stateless, which already causes many problems in fixed networks. Many web content providers therefore use cookies to store some state on a client machine, which is not an elegant solution. State is needed in web browsing, for example, to resume browsing in exactly the same context in which browsing has been suspended. This is an important feature for clients and servers. Client users can continue to work where they left the browser or when the network was interrupted, or users can get their customized environment every time they start the browser.

● **Session management:** WSP introduces sessions that can be **established** from a client to a server and may be long lived. Sessions can also be **released** in an orderly manner. The capabilities of **suspending** and **resuming** a session are important to mobile applications. Assume a mobile device is being switched off – it would be useful for a user to be able to continue operation at exactly the point where the device was switched off. Session lifetime is independent of transport connection lifetime or continuous operation of a bearer network.

 Capability negotiation: Clients and servers can agree upon a common level of protocol functionality during session establishment. Example parameters to negotiate are maximum client SDU size, maximum outstanding requests, protocol options, and server SDU size.

10.3.7 Wireless markup language

Wireless Markup Language (WML), based on XML, is a now-obsolete markup language intended for devices that implement the Wireless Application Protocol (WAP) specification, such as mobile phones. It

provides navigational support, data input, hyperlinks, text and image presentation, and forms, much like HTML (HyperText Markup Language). It preceded the use of other markup languages now used with WAP, such as HTML itself, and XHTML (which are gaining in popularity as processing power in mobile devices increases).

10.3.8 WML Script

WMLScript (Wireless Markup Language Script) is the client-side scripting language of WML (Wireless Markup Language). A scripting language is similar to a programming language, but is of lighter weight. With WMLScript, the wireless device can do some of the processing and computation. This reduces the number of requests and responses to/from the server.

This chapter will give brief description of all the important WML Script components.

WML Script Components

WML Script is very similar to Java Script. WML Script components have almost similar meaning as they have in Java Script. The WML Script program components are summarized here.

WML Script Operators

WML Script supports following type of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Check for complete detail of The WML Operators.

WML Script Control Statements

Control statements are used for controlling the sequence and iterations in a program.

Statement	Description
if-else	Conditional branching
for	Making self-incremented fixed iteration loop

while	Making variable iteration loop
break	Terminates a loop
continue	Quit the current iteration of a loop

Check for complete detail of WML Script Control Statements.

WML Script Functions

The user-defined functions are declared in a separate file having the extension .wmls. Functions are declared as follows –

```
function name (parameters) {
  control statements;
  return var;
}
```

The functions used are stored in a separate file with the extension .wmls. The functions are called as the filename followed by a hash, followed by the function name –

```
maths.wmls#squar()
```

WML Scripts Standard Libraries

The are six standard libraries totally. Here is an overview of them –

- **Lang –** The Lang library provides functions related to the WMLScript language core.

  Example Function – abs(),abort(), characterSet(),float(), isFloat(), isInt(), max(), isMax(), min(), minInt(), maxInt(), parseFloat(), parseInt(), random(), seed()

- **Float –** The Float library contains functions that help us perform floating-point arithmetic operations.

  Example Function – sqrt(), round(), pow(), ceil(), floor(), int(), maxFloat(), minFloat()

- **String –** The String library provides a number of functions that help us manipulate strings.

  Example Function – length(), charAt(), find(), replace(), trim(), compare(), format(), isEmpty(), squeeze(), toString(), elementAt(), elements(), insertAt(), removeAt(), replaceAt()

- **URL –** The URL library contains functions that help us manipulate URLs.

  Example Function – getPath(), getReferer(), getHost(), getBase(), escapeString(), isValid(), loadString(), resolve(), unescapeString(), getFragment()

- **WMLBrowser –** The WMLBrowser library provides a group of functions to control the WML browser or to get information from it.

  Example Function – go(), prev(), next(), getCurrentCard(), refresh(), getVar(), setVar()

- **Dialogs –** The Dialogs library Contains the user interface functions.

  Example Function – prompt(), confirm(), alert()

WML Scripts Comments

There are two types of comments in WMLScript –

- **Single-line comment –** To add a single-line comment, begin a line of text with the // characters.
- **Multi-line comment –** To add a multi-line comment, enclose the text within /* and */.

These rules are the same in WMLScript, JavaScript, Java, and C++. The WMLScript engine will ignore all comments. The following WMLScript example demonstrates the use of comments –

```
// This is a single-line comment.

/* This is a multi-line comment. */

/* A multi-line comment can be placed on a single line. */
```

WML Script Case Sensitivity

The WMLScript language is case-sensitive. For example, a WMLScript function with the name WMLScript Function is different from wmlscript function. So, be careful of the capitalization when defining or referring to a function or a variable in WMLScript.

Whitespaces in WMLScript

Except in string literals, WMLScript ignores extra whitespaces like spaces, tabs, and newlines.

WML Script Statement Termination by Semicolons

A semicolon is required to end a statement in WMLScript. This is the same as C++ and Java. Note that JavaScript does not have such requirement but WML Script makes it mandatory

10.3.9 Wireless telephony application

WTA is a collection of telephony specific extensions for call and feature control mechanisms, merging data networks and voice networks.

It is an Extension of basic WAE application model with following features

–network model for interaction

- client requests to server
- event signaling: server can push content to the client

– event handling

- table indicating how to react on certain events from the network
- client may now be able to handle unknown events

–telephony functions

- some application on the client may access telephony functions

WTAI (Wireless Telephony Application Interface) includes:

–Call control

–Network text messaging

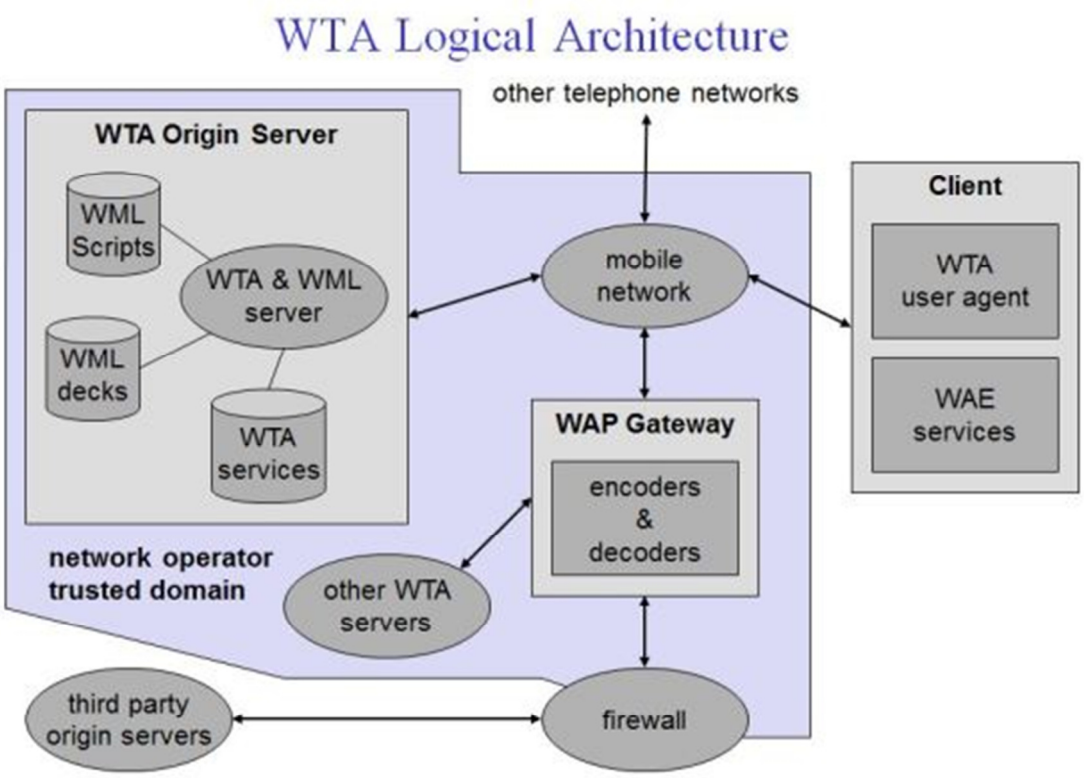
–Phone book interface

–Event processing

Security model: segregation

–Separate WTA browser

WTA logical architecture



The client is connected via a mobile network with a **WTA server**, other telephone networks and a **WAP gateway.**

A WML user agent running on the client is not shown here.

The client may have voice and data connections over the network.

Other origin servers can be connected via the WAP gateway.