ProSeminar "Mobilfunksysteme"
FAU Erlangen-Nürnberg
Dr. Stephan Kindermann

Wireless Application Protocol (WAP)

(Ausarbeitung zum Vortrag am 22. Februar 2000)

Stefan Büttcher
(Matrikel-Nr. 1861351)
Sieglitzhofer Str. 19
91054 Erlangen

Inhaltsübersicht

1. Voraussetzungen / Geschichte

Die Entwicklung des Wireless Application Protocols (WAP) nahm seinen Anfang in einer Initiative von Motorola, Nokia, Ericsson und Phone.com im Sommer 1997. WAP stellt den allerersten Versuch dar, den Internet-Zugriff für mobile Kommunikation zu standardisieren. Inzwischen gehören dem WAP-Forum mehr als 120 Telekommunikations-Unternehmen an, darunter auch Intel, Siemens und die Deutsche Telekom.
Ziel des Forums war und ist es, eine Standardisierung für die Nutzung des Internets mit mobilen Terminals zu schaffen, die weitgehend unabhängig ist von der Umgebung, in der die Terminals benutzt werden. Im Einzelnen bedeutet das, dass WAP unabhängig ist

- von der Art des Mobiltelefons, in dem es betrieben wird (herstellerunabhängig),

- vom zugrundeliegenden Wireless Service als Trägerdienst (z.B. SMS, USSD, GPRS),

- vom Mobilfunkstandard, mit dem es betrieben wird (z.B. GSM, UMTS),

- von der Art des Eingabemediums (z.B. Keyboard, Touch-Screen).

Das bisherige Problem der mangelnden Standardisierung durch viele Hersteller, die ohne Absprache ihre jeweils eigenen "Standards" entwickeln, soll damit aus der Welt geschafft werden. Die Mobilfunkindustrie erhofft sich durch die Einführung von WAP ein starkes Wachstum im Marktsegment der Mobile Information Services, das vor WAP nicht in dem erhofften Umfang stattgefunden hat.

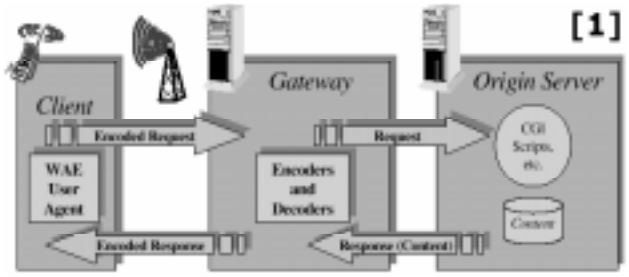
Da das Ziel von WAP die Integration mobiler Kommunikation in das Internet ist, ist WAP in seiner inneren Struktur der jeweils vergleichbaren Internet-Struktur sehr ähnlich, wenn nicht sogar identisch mit dieser.

Obgleich es zum Zeitpunkt erst wenige WAP-fähige Handys auf dem Markt gibt, lässt sich für WAP ein starker Aufwärtstrend feststellen, den man auch mit "Hype" bezeichnen könnte. Siemens und Yahoo haben erst kürzlich ein langfristiges Abkommen zur Vermarktung der Siemens-WAP-Lösungen geschlossen, und sogar Microsoft ist endlich auf den fahrenden Zug aufgesprungen. WAP-Erweiterungen für den Internet-Information-Server werden also wahrscheinlich auch nicht mehr lange auf sich warten lassen.
Nur die Presse scheint noch nicht ganz bemerkt zu haben, dass es wieder etwas Neues gibt, mit dem Zeilen und Seiten gefüllt werden und Leser zufriedengestellt werden können. Doch auch das ist vermutlich nur noch eine Frage von Wochen. Alles, was dann noch fehlt, sind die Benutzer...

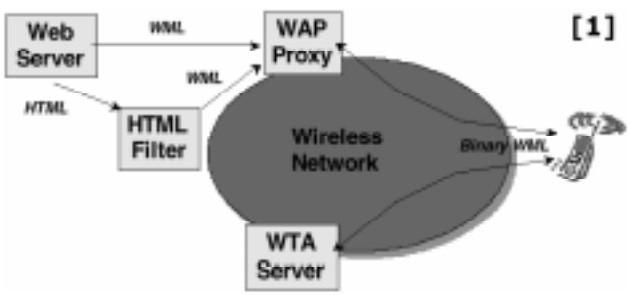
2. Grundsätzliches zur Architektur von WAP

2.1. Das Client/Server-Modell

Das Kommunikationsmodell von WAP zeichnet sich vor allem durch seine Ähnlichkeit mit den verschiedenen Internet-/WWW-Standards aus: Der User-Agent ("Microbrowser") des Mobile Clients fordert eine Ressource an, die durch einen Internet-URL spezifiziert wird. Die Antwort des Gateways bzw. des Internet-Servers erfolgt als WML-Code, konform mit den bestehenden Internet-Standards als "Content-type: text/wml".

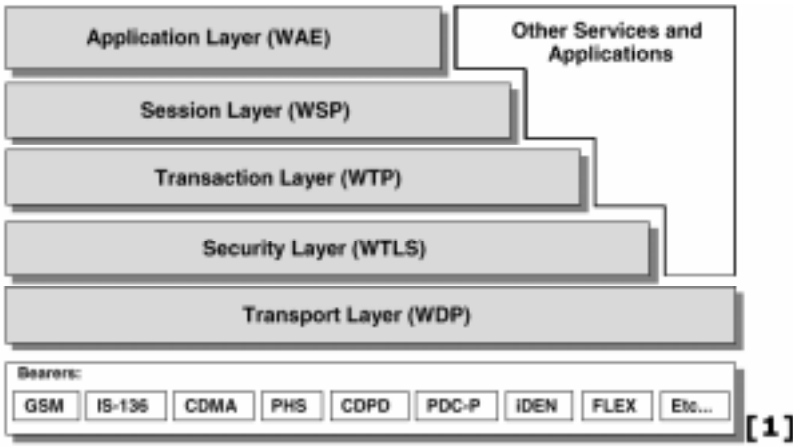


Grundsätzlich bestehen nun zwei ver-schiedene Möglichkeiten der Kommu-nikation.
Möglichkeit 1: Der Microbrowser im Handy des Mobikfunkteilnehmers fordert eine Ressource von einem gewöhnlichen Web-Server an, in der Regel also ein HTML-Dokument. Der WAP-Gateway, an den die Anfrage zunächst gesendet wird, wandelt sie in ein gültiges HTTP-Request um und leitet sie an den Web-Server weiter. Dieser liefert als HTTP-Response das gewünschte Dokument oder einen Error-Code. Der Gateway übersetzt nun die Antwort des Servers und gibt sie als WML-Dokument an den Client weiter.
Möglichkeit 2: Die angeforderte Ressource liegt auf einem WAP Application Server oder einem WAP-fähigen Web-Server. In diesem Fall ist kein Gateway zwischengeschaltet, oder er erscheint als nicht-existent. Die Kommunikation wird in beide Richtungen unverändert weitergeleitet, und der Server liefert die gewünschten Informationen sofort in einem durch WAP-spezifizierten Format, z.B. WML oder WBMP.



In dem rechts zu sehenden Beispiel kommuniziert der Client mit zwei Servern: Einem Web Server und dem Wireless Telephony Application (WTA) Server. Der WTA Server, der dem Client Zugriff auf die Funktionen im Zusammenhang mit der Infrastruktur des Netzbetreibers ermöglicht, ist ein nativer WAP-Server, so dass die Kommunikation direkt erfolgen kann. Die Verbindung mit dem Web Server erfolgt über einen WAP Proxy oder Gateway, der die Datenströme ggf. übersetzt und in beide Richtungen weiterleitet.
Problematisch ist hierbei, dass sich nicht alle Daten, die der Web-Server liefert, einfach so übersetzen lassen, weil sie entweder den zur Verfügung stehenden Speicher des Clients oder seine Anzeigefähigkeiten bei weitem übersteigen.
HTTPS (vormals SSL) wird bisher ebenfalls nicht unterstützt. Gesicherte Kommunikation ist also nur innerhalb des Wireless Networks des jeweiligen Betreibers oder mit speziellen WAP Application Servers möglich.

2.2. Der WAP Protocol Stack



[1]

Wireless Application Environment (WAE)

WAE ist eine Allzweck-Umgebung und basiert auf einer Kombination von WWW- und Mobilfunk-Standards. In der WAE enthalten ist der Microbrowser mit folgenden Fähigkeiten: Wireless Markup Language (WML) zur Darstellung von Inhalten auf dem Display des Mobiltelefons, WMLScript zum Ausführen kleiner Programme im Microbrowser, Wireless Telephony Application (WTA) Services und Interfaces (WTAI) für diverse Funktionen zur Infrastruktur des Netzbetreibers und verschiedene Content-Formate, wie z.B. Grafiken oder Kalender.

Wireless Session Protocol (WSP)

Das WSP stellt dem Application Layer verbindungsorientierte (WTP-basierte) und verbindungslose, unsichere (WDP-basierte) Dienste zur Verfügung. Es ist konzipiert für den Einsatz von "Browsing Applications" Trägerdiensten mit geringer Bandbreite und relativ großen Wartezeiten.

Wireless Transaction Protocol (WTP)

Das WTP stellt drei Arten von Diensten zur Verfügung: unreliable one-way Requests (ohne Bestätigung), reliable one-way Requests (mit Bestätigung) und reliable two-way Request-Reply Transactions (mit Bestätigung).

Wireless Transport Layer Security (WTLS)

WTLS basiert auf SSL und bietet Funktionen zur Datenintegrität (die empfangenen Daten wurden auf ihrem Weg nicht verändert), Privatsphäre (die übermittelten Daten können auf ihrem Weg nicht mitgelesen werden) und Authentizität (der angebliche Absender stimmt mit dem tatsächlichen überein) sowie Schutzfunktionen gegen Denial-of-Service-Attacken.

Wireless Datagram Protocol (WDP)

Das WDP stellt die Verbindungsstelle zum verwendeten Trägerdienst dar und stellt sicher, dass die jeweiligen Dienste des WAP in den höhergelegenen Protokollschichten unabhängig vom aktuellen Bearer (vgl. Diagramm) genutzt werden können.

3. Wireless Application Environment (WAE)

3.1. Wireless Markup Language (WML)

Bei der Betrachtung von WML fällt die große Ähnlichkeit zu HTML auf, was aber nicht weiter wundersam ist bei einer Sprache, die auf XML basiert. Auch das verwendete URL/URI-Schema ist identisch mit dem von HTTP/HTML.

Grundgerüst eines WML-Dokuments [2]

```
<?xml version="1.0"?>
<!DOCTYPE WML PUBLIC "-//WAPFORUM//DTD WML 1.0//EN"
 "http://www.wapforum.org/DTD/wml.xml">
<WML>
   ...
</WML>
```

WML besteht aus sog. "Decks". Jedes dieser Decks enthält die zugrunde-liegende XML-Version und den Doku-mententyp. Des Weiteren muss jedes WML-Deck eine oder mehrere WML-Cards enthalten.

Die Cards weisen wiederum große Ähnlichkeit mit den aus HTML bekannten Formularen auf. Sie enthalten Eingabeelemente ("Input", "Select"), die der Benutzer des Microbrowsers ausfüllen und so mit dem jeweiligen Server kommunizieren kann: Beispiele dafür sind Eingabezeilen, Auswahllisten und Checkboxen.

Beispiel für eine WML-Card [2]

```
<CARD NAME="eCard">
  <DO TYPE="ACCEPT">
     <GO URL="/submit?N=$(N)&S=$(S)"/>
  </DO>
  Enter name: <INPUT KEY="N"/>
  Choose speed:
  <SELECT KEY="S">
    <OPTION VALUE="0">Fast</OPTION>
    <OPTION VALUE="1">Slow</OPTION>
  </SELECT>
</CARD>
```

```
<FORM NAME="eCard" ACTION="/submit" METHOD="POST">
  Enter name:
  <INPUT TYPE=TEXT NAME="N"><BR>
  Choose speed:
  <SELECT NAME="S">
    <OPTION VALUE="0">Fast
    <OPTION VALUE="1">Slow
  </SELECT>
```

Das HTML-Pendant (links) und die obige WML-Card sehen sich im Quelltext zum Verwechseln ähnlich.

Das Einbinden von Bildern in WML ist ebenso einfach wie das Einbinden von Bildern in HTML und gehorcht den gleichen syntaktischen Regeln.
Mit WML lassen sich also beinahe alle aus HTML bekannten Layout-Beschreibungen umsetzen (abgesehen von dynamischem Positionieren, dynamischen Inhalten, u.ä.). Problematisch ist lediglich die Darstellung des durch WML beschriebenen Inhalts auf dem Display des Handys, wegen der unzureichenden Größe, Auflösung und Farbtiefe des Displays. Fotorealistische Abbildungen werden wir wohl in nächster Zeit noch nicht im Display eines Mobiltelefons sehen. Aktienkurse, kleine Charts, Terminplaner und reine Texte dafür wohl schon.

Um bei der Übertragung vom Application Server oder Gateway zum Microbrowser Zeit und Speicherplatz zu sparen, wird der WML-Code nicht als Text übertragen, sondern zuerst compiliert. Das Ergebnis ist ein WML-Bytecode, in dem die üblichen WML-Tokens (z.B. "<card>" zum Beginn einer neuen Card, "
" als Zeichen für einen Zeilenumbruch) durch bestimmte Zahlen ersetzt werden und der im WML-Dokument vorkommende Text ggf. mit Hilfe von Stringtables komprimiert wird.

Beispiel für WML-Bytecode-Darstellung [3]

WML-Deck in Plaintext

```
<wml><card id="abc" ordered="true">
  <p>
    <do type="accept">
      <go href="http://xyz.org/s"/>
    </do>
    X: $(X) <br/>
    Y: $(&#x59;) <br/>
    Enter name: <input type="text" name="N"/>
  </p>
</card></wml>
```

WML-Bytecode (in Hex-Darstellg.)

```
01 04 6A 04 'X 00 'Y 00
7F E7 21 03 'a 'b 'c 00
33 01 20 E8 38 01 AB 4B
03 'x 'y 'z 00 88 03 's
00 01 03 '  'X ': '  00
82 00 26 03 '  'Y ': '
00 82 02 28 03 '  'E 'n
'n 't 'e 'r '  'n 'a 'm
'e ': '  00 AF 48 18 03
'N 00 01 01 01
```

Übersetzungstabelle

01	XML Version 1.1	AB	<go>	82	Variablen-Referenz
04	WML 1.1 Public ID	4B	href="http://"	02	Variablen-Offset 2
6A	Charset UTF-8	03	Inline-String folgt	26	
04	Länge der Stringtable	"xyz" 00	Inline-String	03	Inline-String folgt
"X" 00 "Y" 00	Stringtable	88	".org/"	" Enter name: " 00	Inline-String
7F	<wml>	03	Inline-String folgt	AF	<input>
E7	<card>	"s" 00	Inline-String	48	type="text"
55	id=	01	Ende (von <go>)	21	name=
03	Inline-String folgt	01	Ende (von <do>)	03	Inline-String folgt
"abc" 00	Inline-String	03	Inline-String folgt	"N" 00	Inline-String
33	ordered="true"	" X: " 00	Inline-String	01	Ende (der Input-Attribute)
01	Ende (der <card>-Attribute)	82	Variablen-Referenz	01	Ende (von <p>)
60	<p>	00	Variablen-Offset 0	01	Ende (von <card>)
E8	<do>	26	 	01	Ende (von <wml>)
38	type="accept"	03	Inline-String folgt		
01	Ende (der <do>-Attribute)	" Y: " 00	Inline-String		

Die Bytecode-Darstellung des WML-Decks wird dann vom Microbrowser interpretiert. Diese Interpretation duch den Browser sieht beispielsweise so aus:

Darzustellendes WML-Deck

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
          "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
 <card id="first" title="Demo Application">
  <do type="accept">
   <go href="#next"/>
  </do>
  <p>
   <br/>Druecken Sie "YES", um fort- zusetzen.
  </p>
 </card>
 <card id="next" title="Willkommen!">
  <p>
   Vorname:  <input type="text" name="first"/>
   Nachname: <input type="text" name="last"/>
  </p>
 </card>
</wml>
```

erste Seite | **nach dem ersten YES-Druck** | **nach zweitem YES und Eingabe der Daten**

3.2. Wireless Script Language (WMLScript)

Genau wie bei WML, das analog zu HTML entwickelt wurde, haben die Entwickler von WMLScript sich bei ihrem Vorgehen JavaScript zum Vorbild genommen. Mit WMLScript können kleine Programme und Funktionen geschrieben werden, mit dem Ziel, eine sinnvolle Erweiterung zu WML zu schaffen und gleichzeitig durch bestimmte Mechanismen die zur Verfügung stehende Bandbreite des Netzes besser zu nutzen.
Im Gegensatz zu JavaScript ist WMLScript jedoch - wie Java - Bytecode-basiert und wird im Handy von einer VirtualMachine interpretiert. [8]

Die Standard Libraries von WMLScript umfassen:

- oft benötigte mathematische Funktionen,
- grundlegende Funktionen zur String-Verarbeitung,
- URL-bezogene Funktionen,
- Funktionen zum Browser-Interface,
- einfache Dialog-Interfaces,
- Software-Gleitkomma-Arithmetik. [2]

Innerhalb von WML-Decks kann WMLScript zum Beispiel dazu benutzt werden, um Feldinhalte zu überprüfen, bevor Sie abgeschickt werden und so möglicherweise schon im Vorfeld falsche Eingaben zu erkennen und den Benutzer darauf hinzuweisen (geringerer Traffic), kleinere Berechnungen durchzuführen oder auf die gerätespezifischen Funktionen des Mobile Phones zuzugreifen.

Beispiel für ein WMLScript-Programm [2]

```
function currencyConvertor(currency, exchRate) {
  return currency * exchRate;
}
function myDay(sunShines) {
  var result;
  var myStr = "Not So Good";
  if (sunShines) {
    result = String.subString(myStr, 7, 4);
  } else {
    result = myStr;
  };
  return result;
}
```

Bei der Betrachtung des Codes fällt die Ähnlichkeit mit Java/ JavaScript/C++ sofort ins Auge. Tatsächlich ist die Syntax von WMLScript bis auf wenige Ausnahmen identisch mit der von JavaScript.

Ein Beispiel hierfür ist der Aufruf von Objekt-Methoden: Während hier in JavaScript wie in (fast) jeder anderen Sprache vorgegangen wird, handelt es sich in WMLScript formal nicht um eine Methode des jeweiligen Objekts, sondern um eine Methode der Klasse (im Beispiel **string**), der als erster Parameter das jeweilige Objekt angegeben wird, mit dem gearbeitet werden soll.

Methodenbezeichner und Funktionen sind ebenfalls nicht komplett gleich, weisen jedoch in allen Fällen eine gewisse Ähnlichkeit zueinander auf (z.B. "subString", "substr").

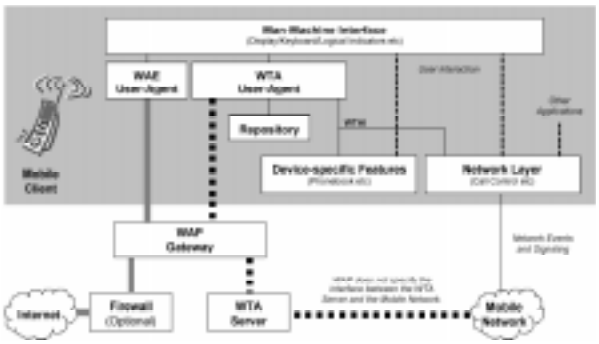
Das JavaScript-Äquivalent

```
function currencyConvertor(currency, exchRate) {
  return currency * exchRate;
}

function myDay(sunShines) {
  var result;
  var myStr = "Not So Good";
  if (sunShines) {
    result = myStr.substr(7, 4);
  } else {
    result = myStr;
  }
  return result;
```

3.3. Wireless Telephony Application (Interface) (WTA/WTAI)

Beim WTA-User-Agent im Mobiltelefon handelt es sich im Wesentlichen um einen ganz normalen WML-Browser, der allerdings um einige netzbezogene Fähigkeiten erweitert ist, zum Beispiel das das Aussenden und Annehmen von Anrufen.



Grobübersicht: WTA-Architektur [4]

Der Zugriff des WTA-Clients auf die netzeigenen Dienste geschieht über den WTA-Server, dessen Ressourcen wie bei einem Web-Server über URL's adressiert werden. Um die zur Verfügung stehende Bandbreite sinnvoll zu nutzen, verfügt das WTA User Agent über ein Repository ("Behälter"), eine Art Cache, in dem oft angeforderte Daten abgelegt werden. WTA stellt mit seinen verschiedenen Diensten die Verbindung zwischen WAP und primitiven Netzfunktionen dar, zum Beispiel das Entgegennehmen eingehender Rufe, das Abfragen bestimmte Netz-Parameter oder das Versenden bzw. Empfangen netzinterner Textnachrichten an andere bzw. von anderen Clients.

Da ein WTA-Dienst verschiedene WTAI-Funktionen aufrufen kann, die den Zugriff auf lokale Funktionen, wie zum Beispiel das Aussenden von Mobile Originating Calls, ermöglichen, muss sichergestellt werden, dass nur WTA-Dienste ausgeführt werden, die vom Handy-Besitzer zugelassen wurden. Wie dieses Problem der User-Permissions vom Netzbetreiber gelöst wird, ist in WAP nicht festgelegt.

Prinzipiell werden jedoch drei Arten von Permissions unterschieden [4] :

1. *Blanket Permission*
   Der Benutzer gibt der jeweiligen WTAI-Funktion eine Blanko-Erlaubnis, üblicherweise zum Zeitpunkt der Einrichtung. Die Funktion kann danach jederzeit ohne weitere Rückfragen ausgeführt werden, der Benutzer kann ihr diese Permission jedoch zu jedem beliebigen Zeitpunkt wieder entziehen.

2. *Session Permission*
   Der Benutzer gibt die Permission für die Dauer einer bestimmten Session, während der die Funktion uneingeschränkt arbeiten kann. Nach Beendigung der Session geht die Permission verloren. Der Benutzer kann die Erlaubnis jederzeit entziehen.

3. *Single Action Permission*
   Die Permission gilt für einen einzigen Aufruf der Funktion. Ist die Funktion beendet, wird die Permission gelöscht und muss für den nächsten Funktionsaufruf neu vergeben werden.

3.4. WAP Content Types

Außer den bereits beschriebenen Content Types (WML für die Darstellung von Texten/Dokumenten und WMLScript zum Ausführen kleiner Programme im Speicher des Terminals) definiert WAP einige weitere Typen, die ich im Folgenden kurz beschreiben werde:

Wireless Markup Language
text/wml (.wml)

WML dient dem Darstellen textueller Inhalt und wurde bereits weiter oben ausführlich beschrieben.

WML-Bitmap (WBMP)
image/x-wap.wbmp (.wbmp)

WML-Bitmaps werden benutzt, um grafische Inhalte innerhalb von WML-Decks darzustellen. Momentan sind die durch WBMP beschriebenen Grafiken durch den Stand der Technik (nämlich die Displays der Mobiltelefone) auf eine Farbtiefe von 1 Bit beschränkt. Theoretisch lassen sich aber auch True-Color-Grafiken mit WBMP darstellen. Optional können die Bitmap-Daten einer Grafik auch komprimiert werden, mögliche Kompressionsverfahren sind aber noch nicht definiert.

Portable Network Graphics
image/png (.png)

PNG ist zwar kein WAP-eigenes Format, dennoch sind alle Microbrowser, die der WAE-Spezifikation gehorchen, in der Lage, PNG-Bilder zu erkennen und - falls das Display das zulässt - darzustellen.

vCard
text/x-vcard (.vcf)

Die WAP-vCard ist ein bereits bestehender Industrie-Standard zum Austausch von Telefonbucheinträgen und Business Cards, der von WAP übernommen wurde. vCards werden in der Regel durch einfache WDP-Datagramme ausgetauscht.

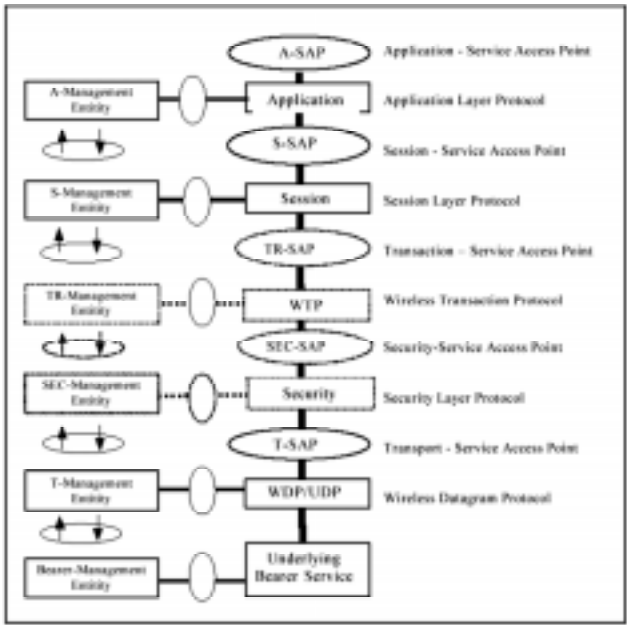
vCalendar
text/x-vcal (.vcs)

Wie die vCard ist vCalendar ein von der Industrie übernommener Standards, diesmal zum Austausch von Termin- und Kalender-Informationen. Auch hier geschieht der Austausch in aller Regel durch einfache Datagramme.

Unter den Namen der Content-Types stehen jeweils kursiv die MIME-Media-Type-Bezeichnungen und die gebräuchlichen Datei-Endungen.

4. Wireless Protocols

4.1. Wireless Session Protocol (WSP)



Überblick über die Architektur [5]

WSP ist ein Session-orientiertes Protokoll zur Kommunikation zwischen Client und Server oder Proxy. Wie aus dem Diagramm zu sehen, basiert es auf den Transaction und Datagram Services von WAP. Wie alle Protokolle des WAP ist auch WSP auf die speziellen Erfordernisse eines Mobile Phones ausgerichtet, nämlich geringe Netz-Bandbreite und geringe Hardware-Kapazität (Prozessor /Arbeitsspeicher).
Generell stellt WSP folgende Dienste zur Verfügung: ordungsgemäßer Auf- und Abbau einer reliable Session zu einem Server, bidirektionaler Datenaustausch zwischen Client und Server mit geeigneten Codierungsverfahren, die Möglichkeit, eine Session vorübergehend zu unterbrechen und zu einem späteren Zeitpunkt wieder aufzunehmen (suspend/resume), und Mechanismen zum Beenden einer Session.
Die eigentlichen Übertragungs-Funktionen (MethodInvoke, MethodResult und Push) werden sowohl connection-oriented als auch connectionless angeboten. Im ersten Fall muss vor der Übertragung der Informationen gemäß dem Request-Response-Schema eine Verbindung zwischen Client und Server aufgebaut und später wieder beendet werden. Bei den Connectionless Services wird keine Verbindung aufgebaut, sondern der Client sendet einfach ein MethodRequest an den Server, in der Hoffnung, dass dieser die Anfrage bekommt, verarbeitet und die gewünschten Daten liefert.
Abgesehen von diesen Besonderheiten, ist WSP sehr ähnlich zu HTTP, analog zur Byte-Codierung bei WML werden aber auch hier die einzelnen Requests/Responses nicht als Plaintext, sondern Byte-codiert übertragen, um Zeit zu sparen. Zum schonenden Umgang mit den Akkus und der Netzkapazität des Betreibers kann eine Session, wenn gerade keine Daten übertragen werden, die benutzten Ressourcen freigeben und in einen Suspend-Mode übergehen, aus dem es später mit Hilfe des Session Re-Establishment Protocols wieder zurückgerufen werden, ohne den Overhead eines komplett neuen Session-Aufbaus mit sich zu bringen.
Eine weitere Besonderheit von WSP ist die Möglichkeit, des Servers, zu einem beliebigen Zeitpunkt einer laufenden Session unaufgefordert ("unsolicited") mittels der Push-Funktion Daten an den Client zu versenden (und damit das Client-Server-Konzept ein wenig auf den Kopf zu stellen). Man unterscheidet zwischen Confirmed-Pushs und Non-Confirmed-Pushs, die von WAP-Client jeweils entweder bestätigt werden (confirmed) oder eben nicht (non-confirmed).

Die wichtigsten Connection-oriented Service Primitives von WSP sind:

- *S-Connect:*

Parameter \ Primitive	S-Connect			
	req	*ind*	*Res*	*cnf*
Server Address	M	M(=)	–	–
Client Address	M	M(=)	–	–
Client Headers	O	C(=)	–	–
Requested Capabilities	O	M	–	–
Server Headers	–	–	O	C(=)
Negotiated Capabilities	–	–	O	M(=)

[5]

S-Connect ist für das Session Establishment verantwortlich. Client- und Server-Header sind HTTP-verträgliche Header-Informationen und konstant für die Dauer der Session. Requested- und Negotiated-Capabilities dienen zum Vereinbaren eines Satzes gemeinsamer Funktionen von Server und Client. Client- und Server-Address erklären sich selbst.

- *S-MethodInvoke:*

Parameter \ Primitive	S-MethodInvoke			
	req	*ind*	*res*	*cnf*
Client Transaction Id	M	–	–	M(=)
Server Transaction Id	–	M	M(=)	–
Method	M	M(=)	–	–
Request URI	M	M(=)	–	–
Request Headers	O	C(=)	–	–
Request Body	C	C(=)	–	–

[5]

S-MethodInvoke wird vom WAP-Client benutzt, um ein Request an den Server zu starten. Die Transaction-Ids werden dabei auf beiden Seiten benötigt, um zwischen verschiedenen laufenden Transaktionen zu unterscheiden. Method ist die Methode, mit der die Informationen angefordert werden - entweder Standard-HTTP-Methoden (GET, POST) oder erweiterte Methoden, die beim Aufbau der Session vereinbart wurden. URI, Headers und Body sind jeweils identisch mit ihrem Gebrauch innerhalb von HTTP.

- *S-MethodResult:*

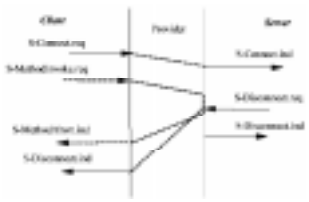
Parameter \ Primitive	S-MethodResult			
	req	*ind*	*res*	*cnf*
Server Transaction Id	M	–	–	M(=)
Client Transaction Id	–	M	M(=)	–
Status	M	M(=)	–	–
Response Headers	O	C(=)	–	–
Response Body	C	C(=)	–	–
Acknowledgement Headers	–	–	O	P(=)

[5]

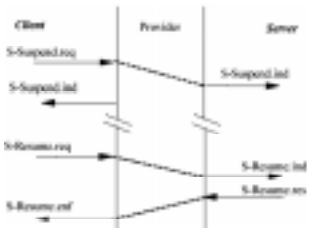
S-MethodRequest kann nur auf ein vorhergehendes S-MethodInvoke folgen und dient der Übertragung der angeforderten Information. Für die Transaction-Ids, die Headers und den Body gilt dasselbe wie oben. Neu sind der Respone-Status-Code, der identisch ist mit dem HTTP-Status-Code, und die Ack-Headers, mit denen zusätzliche Informationen übertragen werden können, die der Provider jedoch nach belieben kürzen oder entfernen darf.

Beispielhafte Abläufe von Kommunikation über WSP [5] :

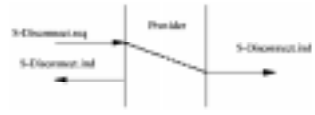
Successful Session establishment



Refused Session establishment

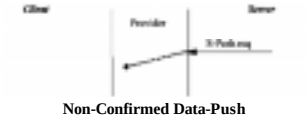


Session Suspension and Resume



Active Session Termination

Confirmed Data Push



Non-Confirmed Data-Push

4.2. Wireless Transaction Protocol (WTP)

Das Wireless Transaction Protocol hat die Aufgabe, dem WSP verschiedene Transaction Services zur Verfügung zu stellen, auf deren Basis das WSP dann arbeiten kann. Diese Transaction Services lassen sich prinzipiell in 3 Klassen unterteilen:

- Class 0: *Unzuverlässige Nachrichten ohne Antwort*
  (unreliable invoke message with no response message, unreliable datagram service)

  Es wird eine Nachricht (Invoke Message) vom Initiator zum Responder gesandt. Der Responder verschickt keine Empfangsbestätigung, so dass der Initiator nicht erfährt, ob seine Nachricht angekommen ist. Die Transaktion endet beim Initiator, sobald das Paket komplett verschickt ist, beim Responder, sobald er es komplett empfangen hat. Die Transaktion kann nicht unterbrochen werden.

- Class 1: *Verlässliche Nachrichten ohne Antwort*
  (reliable invoke message with no response message, reliable datagram service)

  Es wird eine Invoke Message vom Initiator zum Responder gesandt. Der Initiator verschickt ein Acknowledgement. Nach dem Empfang der Nachricht behält der Responder bestimmte Status-Informationen noch für einige Zeit im Speicher, um sie bei eventuellen Re-Transmissions (falls das Ack nicht beim Initiator angekommen ist) wiederverwenden zu können. Auf Seite des Initiators endet die Transaktion, sobald er das Acknowledgement empfangen hat. Ist nach einer bestimmten Zeitspanne noch immer kein Acknowledgement angekommen, verschickt der Initiator die Nachricht erneut. Die Transaktion kann jederzeit abgebrochen werden.

- Class 2: *Verlässliche Nachrichten mit genau einer verlässlichen Antwort*
  (reliable invoke message with exactly one reliable response message, invoke/response transaction service)

  Es wird eine Invoke Msg vom Initiator zum Responder gesandt. Der Responder antwortet mit genau einer Response Message, die das Acknowledgement der ursprünglichen Nachricht (z.B. durch Angabe der Transaction-Id) impliziert. Bei Empfang der Antwort sendet der Initiator ein Acknowledgement an den Responder. Ist die User-Acknowledgement-Funktion aktiviert, muss der Benutzer am Responder den Eingang der Nachricht zunächst bestätigen, bevor die Antwort generiert wird.

Im Gegensatz zum WSP werden vom WTP keine Connections mehr aufgebaut, da es auf dieser Ebene bereits viel zu viel Overhead produzieren und die kostbare Bandbreite des Mobilfunknetzes verschwenden würde. Die Übertragung geschieht Paket-basiert [6].

Die Nachrichten werden in sog. Protocol Data Units (PDU) übertragen. Jede PDU besteht aus zwei

Teilen: dem Header und den Daten. Der Header unterteilt sich wiederum in festen und variablen Header. Der feste Header steht für häufig benutzte Parameter und den PDU-Code zur Verfügung.

Folgende PDU-Typen sind momentan definiert:

PDU Type	PDU Code
NOT ALLOWED	0x00
Invoke Message	0x01
Result Message	0x02
Acknowledgement	0x03
Abort	0x04
Segmented Invoke	0x05
Segemnted Result	0x06
Negative Ack	0x07

Das erste Bit des festen Header gibt an, ob die PDU über einen variablen Header verfügt. Die Länge des festen Headers wird durch die PDU Type festgelegt (s. links).
Im variablen Teil des Headers werden die seltener benötigten Parameter festgehalten. Variable Parameter werden in Transport Information Items (TPI) übertragen.
Das erste Bit eines jeden TPI gibt an, ob es sich um das letzte TPI handelt. Jedes TPI hat einen eigenen Eintrag, der angibt, wie groß es ist.

Bei der Übertragung von "multi-octet-integers" (Integer-Werten, die breiter sind als 8 Bit), erfolgt die Übertragung "big-endian", also mit dem Most Significant Byte zuerst.

Beispiele für PDU's und TPI's [6]:

Bit/Octet	0	1	2	3	4	5	6	7
1	CON	PDU Type = Invoke				GTR	TTR	RID
2 3	TID							
4	Version		TIDnew	U/P	RES	RES	TCL	

Invoke PDU (Nachricht senden)

Feld *Beschreibung*

CON Continue Flag
gibt an, ob variabler Header vorhanden ist (1) oder nicht (0)

GTR/TTR Group Trailer / Transmission Trailer
00b: letztes Paket noch nicht erreicht
01b: letztes Paket der Nachricht erreicht
10b: letztes Paket der Paket-Gruppe erreicht
11b: Segmentation nicht unterstützt

RID Re-Transmission Indicator
gibt an, ob es sich um eine Re-Transmission (1) handelt oder nicht (0)

TID Transaction-Identifier
wird benötigt, um jedes Paket eindeutig mit einer Transaktion in Verbindung zu setzen

Version die verwendetee Version des WTP (00b)

TIDNew gibt an, ob die Transaction-Id geändert wurde (1) oder nicht (0)

U/P gibt an, ob der Initiator ein User Ack benötigt (1) oder nicht (0)

RES Reserved

TCL Class
eine der drei Klassen, wie oben definiert, binär codiert

Bit/Octet	0	1	2	3	4	5	6	7
1	CON	TPI Identity = 0x00				0	TPI Length = 0x02	
2	Error Code = 0x02					Bad TPI Identity		
3	First octet of TPI							

Error TPI

Einen Speziallfall unter den TPIs stellt das Option TPI dar. Mit ihm werden spezielle Kommunikationspara-meter festgelegt. Die Parameter innerhalb des Option-TPI gelten für die Dauer der gesamten Transaktion.

Table 29 Encoding of Option TPI

Option	Identity	Description	Comment
Maximum Receive Unit	0x01	This parameter is used by the Initiator to advertise the maximum unit of data in bytes that can be received in the result	
Total Message Size	0x02	This parameter can be sent in the first packet of a segmented message to inform the receiver about the total message size in bytes	Note 1
Delay Transmission Timer	0x03	This parameter can be sent in the Ack PDU when a packet group is acknowledged. The receiver MUST NOT send the next packet group until the specified time has elapsed. The time is in 1/10 seconds.	Note 1
Maximum Group	0x04	This parameter can be used by either transaction party to advertise the maximum group size which can be received. The parameter indicates the maximum size in bytes of data in a single group.	
Current TID	0x05	This parameter may be sent with an Ack PDU when a 3-way-handshake is requested by the server, i.e. the Verify flag is set. The use of the parameter is optional, and the interpretation by the client implementation dependent. When used, the value shall be the value cashed by the server (LastTID).	
No Cached TID	0x06	This parameter may be sent with an Ack PDU when a 3-way-handshake is requested by the server, i.e. the Verify flag is set. The use of the parameter is optional, and the interpretation by the client implementation dependent. When used, the parameter indicated that there is no cashed LastTID	

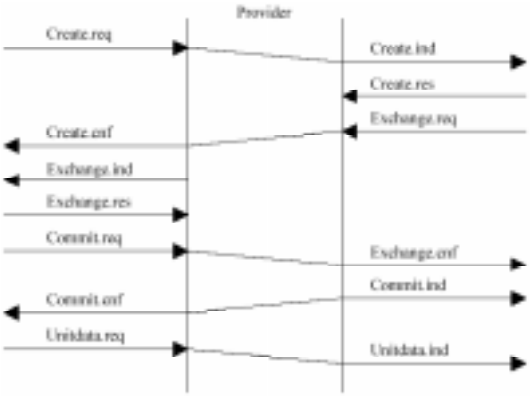
The structure of the Option TPI is illustrated below.

Table 30 Structure of Option TPI

Bit/Octet	0	1	2	3	4	5	6	7
1	CON	TPI Identity				0	TPI Length = N	
2	Option Identity							
3 ... 1+N	Option Value							

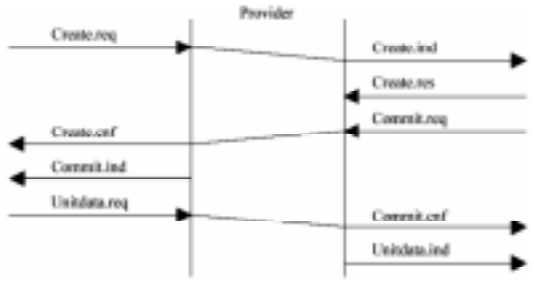
4.3. Wireless Transport Level Security (WTLS)

Der Security Layer von WAP, auf den ich nur kurz eingehen möchte, sorgt für die nötige Sicherheit bei der Datenübertragung. Die von der WTLS zur Verfügung gestellten Funktionen sind jedoch nur auf Übertragungen innerhalb des jeweiligen Netzes anwendbar. *WTLS stellt keine Funktionen für gesicherte Kommunikation mit einem Web-Server oder einer anderen Datenquelle außerhalb des eigenen Mobilfunknetzes zur Verfügung!*
Der Security Layer macht dabei von den üblichen asymmetrischen Verschlüsselungs-Mechanismen Gebrauch, die auch im Internet zum Einsatz kommen. Es handelt sich hier allerdings wiederum um spezielle, den Erfordernissen der Umgebung und der geringen Kapazität des Mobile Phones angepasste Implementierungen.
Um Rechenzeit und Bandbreite zu sparen, ist es mit WTLS zum Beispiel auch möglich, beim Aufbau einer neuen gesicherten Verbindung auf Teile des Handshakes zu Verzichten und die Sicherheitsdaten, wie zum Beispiel den Public Key des Peers, von einer bereits bestehenden Verbindung mit eben diesem zu übernehmen.



Volles Handshake [7]

Zu sehen ist hier der Unterschied zwischen vollem und beschleunigtem Handshake.



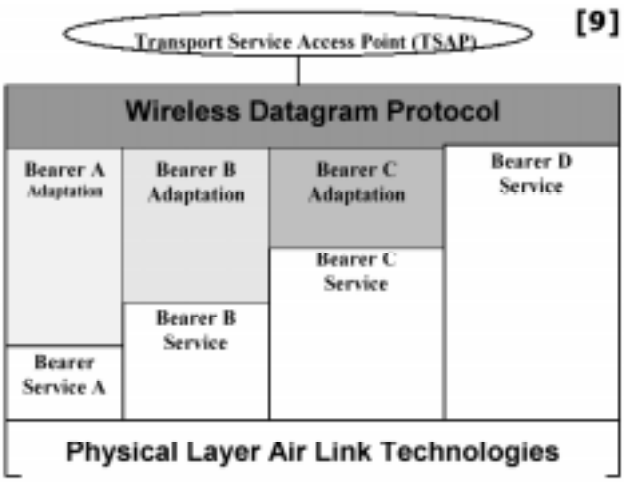
Beschleunigtes Handshake [7]

WTLS stellt zur Sicherung der Daten folgende vier Operationen zur Verfügung, die von den darüberliegenen Protokollschichten genutzt werden können, aber nicht müssen: Digital Signing, Stream Cipher Encryption, Block Cipher Encryption und Public Key Encryption.

Digital Signing	wird mittels einer Einweg-Hash-Funktion erreicht (SHA oder MD5).
Stream Cipher Encryption	ist realisiert durch eine einfache Exklusiv-Oder-Verknüpfung der Daten des zu sendenden/empfangenden Stroms mit Daten aus einem Pseudo-Zufallsgenerator.
Block Cipher Encryption	arbeitet mit ganzen Blöcken, so dass die Länge des Cipher-Textes immer ein ganzes Vielfaches der Blocklänge ist. Mögliche Chiffre-Algorithmen sind IDEA, DES, Triple-DES.
Public Key Encryption	sorgt für die asymmetrische Verschlüsselung, so dass mit einem öffentlich bekannten (Public) Schlüssel chiffrierte Daten nur mit einem geheimen Private Key entschlüsselt werden können. Das übliche Verfahren ist hierbei RSA/RC5-56.

4.4. Wireless Datagram Protocol (WDP)

Das Wireless Datagram Protocol stellt den höheren Protokollschichten konsistente Funktionen zum Versenden und Empfangen von Daten über verschiedene Wireless Protocolls zur Verfügung, so dass es für die Implementierung eines Session-Protokolls oder eines Microbrowsers keine Rolle mehr spielt, ob die Informationen z.B. über SMS oder über USSD verschickt werden.



Der interne Aufbau einer WDP-Implementierung, auf den ich nicht näher eingehen möchte, hängt stark vom Trägerdienst ab, auf den sie aufgesetzt ist. Bisher existieren unter anderem WDP-Implementierungen für folgende Trägerdienste: SMS, USSD, CSD und GPRS (jeweils über GSM, IS-136, CDMA, PDC o.a.).
Je nachdem, auf welchen Bearer aufgesetzt wird, variieren Implementierung und Performance von WDP ganz erheblich, was durch die unterschiedlich hohen Säulen der Bearer Services angedeutet werden soll.

In allen Fällen, in denen der verwendete Trägerdienst, IP-basiert arbeitet, ist es weder nötig noch sinnvoll, mit WDP ein neues Protokoll zur Verfügung zu stellen. Es wird dann einfach das UDP (User Datagram Protocol) verwendet, das ja ebenfalls auf IP aufsetzt.

Beim Übertragen von Daten werden Sender- und Empfänger-Software - genau wie im Internet - eindeutig angesprochen durch ihre Adresse im Netzwerk (z.B. IP-Adresse) und den jeweils verwendeten Port (als 8- oder 16-Bit-Integer).

In WAP/WDP definierte Port-Nummern

Portnummer	Applikation/Protokoll
9200	WAP Connectionless Session Service
	Protokoll: WSP/Datagram
9202	WAP Secure Connectionless Session Service
	Protokoll: WSP/WTLS/Datagram
9201	WAP Session Service
	Protokoll: WSP/WTP/Datagram
9203	WAP Secure Session Service
	Protokoll: WSP/WTP/WTLS/Datagram
9204	WAP vCard
	Protokoll: vCard/Datagram
9206	WAP vCard Secure
	Protokoll: vCard/WTLS/Datagram
9205	WAP vCal
	Protokoll: vCalendar/Datagram
9207	WAP vCal Secure
	Protokoll: vCalendar/WTLS/Datagram

5. Zusammenfassung / Pro&Contra / Ausblick in die Zukunft

Alles in allem scheint es sich bei WAP um einen sinnvollen und erfolgversprechenden Ansatz zu handeln. Für die Einführung und Verwendung von WAP als Mobile Information Standard spricht, dass es bislang noch keinen solchen Standard gibt. Die Größe des WAP-Forums und die Art der Mitglieder legt die Vermutung nahe, dass WAP wirklich **der** Standard wird.
Gleichzeitig ist die Gesamtstruktur von WAP durch seine Ähnlichkeit zu den einschlägigen Internet-Protokollen bestens für die Integration des Handys ins Internet geeignet und auf der anderen Seite durch sinnvolle Änderungen, Streichungen und Ergänzungen hinsichtlich der eingeschränkten Kapazität sowohl des Netzes als auch der Client-Hardware selbst auch sehr gut auf die Bedürfnisse der Benutzer abgestimmt (drastische Verringerung des Overheads, intelligentes Power-Management durch Suspend-Mechanismen).
Als weiterer Pluspunkt ist zu vermerken, dass WAP in der Lage ist, auf (beinahe) jedem beliebigen Trägerdienst zu operieren, und das sogar parallel. Der universale Charakter von WAP ist eine seiner größten Stärken.

Trotz dieser Universalität ergeben sich gewisse Probleme daraus, dass es bisher nur einen Trägerdienst gibt, der wirklich für den Einsatz als WAP-Bearer geeignet ist:

- SMS (Short Message System) ist mit seiner Beschränkung auf 160 Zeichen pro Nachricht nicht geeignet, weil allein der WAP-Overhead schon mehrere SMS-Nachrichten für einen einzigen Request benötigt (SBC ist weltweit der einzige Netzbetreiber, der SMS benutzt);

- CSD (Circuit Switched Data) wird momentan von den meisten Betreibern zur Realisierung von WAP genutzt, problematisch ist hier jedoch der Verbindungsaufbau - bei jedem neuen Aufbau verstreichen 10 (best case, bei komplett digitaler Verbindung) bis 30 Sekunden (worst case, im Fall analoger Modems auf der Strecke), bis die Verbindung steht;

- GPRS (General Packet Radio Service) kommt ohne Dial-Up-Connection aus und verfügt über ausreichend hohe Datenraten für WAP (bis zu 177,2 kbps), ist aber momentan noch nicht als "Mobile Terminated" verfügbar;

- USSD (Unstructured Supplementary Services Data) arbeitet Session-orientiert, es muss also nur ein einziges Dial-Up stattfinden, bei bidirektionaler Kommunikation ist USSD ungefähr 7x so schnell wie SMS; weil alle USSD-Kommandos an das HLR (Home Location Register) des Mobilfunkteilnehmers zurückgeroutet werden, ist auch Roaming kein Problem; USSD wäre also als Trägerdienst für WAP geeignet, zumal es im GSM-Standard enthalten ist und damit auf jedem GSM-Telefon funktioniert.

Ein weiteres Problem stellen die Gateway-Hersteller dar, die im Streit, wer die meisten WAP-Gateways verkauft, die Weiterentwicklung aufhalten.
Die Übersetzung von in HTML verfassten Internet-Seiten nach WML ist bisher nur unzureichend gelöst. Oft - gerade auch bei komplexeren Seiten mit Bildern und Grafiken - ist die Übersetzung gar nicht möglich, weil die Informationen in den Bildern nicht in ein 1-Bit-WBMP konvertiert werden können.

Für die Darstellung nativer WML-Dokumente oder übersetzter Internet-Seiten werden größere Displays, mehr Rechenpower und mehr Speicher benötigt. Das steht der momentan zu beobachtenden Tendenz "kleiner und leichter" entgegen. Die Handys müssten also wieder größer werden. Alternativ wäre auch die Entwicklung völlig neuer Konzepte zur Darstellung von Inhalten denkbar, z.B. Laser-Projektionen. Bis solche Verfahren aber Realität sind, wird sicherlich noch das eine oder andere Jahrzehnt verstreichen.
Gesicherte Datenübertragung vom und ins Internet kann bisher noch nicht realisiert werden. Es gibt zwar einen speziellen Security Layer im WAP-Stack, der sorgt jedoch nur für die notwendige Sicherheit innerhalb des Mobilfunknetzes. Eine Realisierung für HTTPS/SSL in WAP gibt es noch nicht.
Zuguterletzt wird WAP am Anfang sehr teuer sein, gerade bei einem weniger geeigneten Träger wie SMS (allein das Verschicken einer einzigen verlässlichen Nachricht mit dem darauffolgenden Acknowledgement benötigt über SMS selbst bei bereits bestehender Verbindung noch mehr als 5 Sekunden) und großen Datenmengen aus dem Internet werden lange Download-Zeiten benötigt, die entprechenden Einfluss auf die Monatsabrechung des Mobilfunkteilnehmers nehmen.

Ein kurzer Ausblick in die Zukunft:

Die Entwicklung und Einführung der "Zukunftstechnologien" in die Mobilfunknetze wird gerade für WAP ungeheure Konsequenzen haben. Egal, ob GPRS, UMTS oder völlig neue Übertragungstechniken - der maximal mögliche Datendurchsatz liegt deutlich über dem von ISDN. Die im Moment noch sinnvollen und notwendigen Anpassungen der Internet-Protokolle an die spezielle Situation der Handys im Rahmen von WAP werden schon in wenigen Jahren hinfällig sein, da nicht nur die Bandbreite der Mobilfunknetze, sondern auch die Kapazität der in den Handys verwendeten Mikroprozessoren in einer unglaublichen Geschwindigkeit ansteigen.
Das eröffnet völlig neue Welten für die mobile Kommunikation, WAP wird aber wahrscheinlich in keiner von ihnen vertreten sein. Als Übergangslösung durchaus geeignet und sinnvoll, wird es sich jedoch kaum gegen IP durchsetzen können. Auch Notebooks werden immer kleiner und immer leistungsfähiger. Warum also nicht einfach mit dem Notebook surfen, anstatt sich mit einem viel zu kleinen Display und einem viel zu kleinen und unpraktischen Eingabemedium durch das Netz der Netze zu quälen?
Der eigentliche Vorteil von WAP gegenüber den Standard-Protokollen aus der IP-Suite, der sehr geringe Overhead, ist kein Vorteil mehr, wenn mit dem Handy konstante Datenraten von über 100 kbps erreicht werden können. Wenn WAP auf Dauer erhalten bleibt, dann wohl nur als Adaption Layer, um TCP darauf aufzusetzen und die Wireless Bearer Services nutzen zu können.
Vielleicht schwirrten den Erfindern von WAP ja bei der Grund-Idee bereits ähnliche Gedanken im Kopf herum wie mir, und sie haben WAP von Anfang an so nah wie möglich an der IP-Suite gehalten, um es in einigen Jahren einmal einfacher in TCP/IP aufgehen zu lassen.

6. Abkürungsverzeichnis und Literaturangaben

Abkürzungsverzeichnis:

CDMA	Code Division Multiple Access
CGI	Common Gateway Interface
CSD	Circuit Switched Data
DES	Data Encryption Standard
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communication
GTR	Group Trailer (Packet-Group-Ende)
HLR	Home Location Register
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
PDU	Protocol Data Unit
SHA	Secure Hash Standard
SMS	Short Message Service
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TID	Transaction Identifier
TLS	Transport Layer Security
TTR	Transmission Trailer (Übertragungsende)
UDP	User/Unreliable Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USSD	Unstructured Supplementary Service Data
WAE	Wireless Application Environment
WAP	Wireless Application Protocol
WBMP	Wireless Bitmap
WDP	Wireless Datagram Protocol
WSP	Wireless Session Protocol
WTA	Wireless Telephony Applications
WTAI	Wireless Telephony Applications Interface
WTLS	Wireless Transport Layer Security
WTP	Wireless Transaction Protocol
WWW	World Wide Web
XML	Extensible Markup Language

Literaturverzeichnis:

[1] WAP Forum, 1999: "Wireless Application Protocol Architecure Overview"
[2] Flaherty/Dahm, 1998: "Wireless Application Protocol Technical Overview"
[3] WAP Forum, 1999: "Wireless Markup Language Specification"
[4] WAP Forum, 1999: "Wireless Telephony Application Specification"
[5] WAP Forum, 1999: "Wireless Session Protocol Specification"
[6] WAP Forum, 1999: "Wireless Transaction Protocol Specification"
[7] WAP Forum, 1999: "Wireless Transport Layer Security Specification"
[8] WAP Forum, 1999: "WMLScript Language Specification"
[9] WAP Forum, 1999: "Wireless Datagram Protocol Specification"