Protocol (HTTP, Hypertext Transfer Protocol) and language (HTML, Hypertext Markup Language) of the Web have not been designed for mobile applications and mobile devices, thus creating many problems!

- Typical transfer sizes
  - HTTP request: 100-350 byte
  - responses avg. <10 kbytes, header 160 bytes, GIF 4.1kbytes, JPEG 12.8 kbytes, HTML 5.6 kbytes
  - but also many large files that cannot be ignored

- The Web is not a file system
  - web pages are not simple files to download
  - static and dynamic content, interaction with servers via forms, content transformation, push technologies, etc.
  - many hyperlinks, automatic loading and reloading, redirecting
  - a single click might have big consequences!

WWW example

Request to port 80:

GET / HTTP/1.0

or:

GET / HTTP/1.1

Host: www.inf.fu-berlin.de

Response from server

HTTP/1.1 200 OK
Date: Wed, 30 Oct 2002 19:44:26 GMT
Server: Apache/1.3.12 (Unix) mod_perl/1.24
ETag: "2d8190-2322-3dbfdbaf"
Accept-Ranges: bytes
Content-Length: 8994
Connection: close
Content-Type: text/html

<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>FU-Berlin: Institut für Informatik</title>
<base href="http://www.inf.fu-berlin.de">
<link rel="stylesheet" type="text/css" href="http://www.inf.fu-berlin.de/styles/homepage.css">
<!--script language="JavaScript" src="fuinf.js"-->
<!--/script-->
</head>
<body onResize="self.location.reload();">
...
HTTP 1.0 and mobility I

- Characteristics
  - stateless, client/server, request/response
  - needs a connection oriented protocol (TCP), one connection per request (some enhancements in HTTP 1.1)
  - primitive caching and security

- Problems
  - designed for large bandwidth (compared to wireless access) and low delay
  - big and redundant protocol headers (readable for humans, stateless, therefore big headers in ASCII)
  - uncompressed content transfer
  - using TCP
    - huge overhead per request (3-way-handshakes) compared with the content, e.g., of a GET request
  - slow-start problematic
  - DNS lookup by client causes additional traffic

HTTP 1.0 and mobility II

- Caching
  - quite often disabled by information providers to be able to create user profiles, usage statistics, etc.
  - dynamic objects cannot be cached
    - numerous counters, time, date, personalization, ...
  - mobility quite often inhibits caches
  - security problems (authentication between client and server)
  - today: many user customized pages, dynamically generated on request via CGI, ASP, ...

- POSTing (i.e., sending data to a server)
  - can typically not be buffered, very problematic if currently disconnected

HTML and mobile devices

- HTML
  - designed for computers with “high” performance, color high-resolution display, mouse, hard disk
  - typically, web pages optimized for design, not for communication

- Mobile devices
  - often only small, low-resolution displays, very limited input interfaces
    - small touch-pads, soft-keyboards, low performance, ...

- Additional “features”
  - animated GIF, Java Applets, Frames, ActiveX Controls, Shockwave, movie clips, audio, ...
  - many web pages assume true color, multimedia support, high-resolution and many plug-ins

- Web pages ignore the heterogeneity of end-systems!
  - e.g., without additional mechanisms, large high-resolution pictures would be transferred to a mobile phone with a low-resolution display causing high costs
Approaches for WWW for mobile devices

- Application gateways, enhanced servers
  - simple clients, pre-calculations in the fixed network
  - compression, filtering, content extraction
  - automatic adaptation to network characteristics

- Examples
  - picture scaling, color reduction, transformation of the document format (e.g., PS to TXT)
  - clipping, zooming, etc.
  - headline extraction, automatic abstract generation
  - HDML (handheld device markup language): simple language similar to HTML requiring a special browser
  - HDTP (handheld device transport protocol): transport protocol for HDML, developed by Unwired Planet

- Problems
  - proprietary approaches, require special enhancements for browsers
  - heterogeneous devices make approaches more complicated

Some new issues that might help mobility?

- Push technology
  - real pushing, not a client pull needed, channels, etc.

- HTTP1.1
  - client/server use the same connection for several request/response transactions ("persistent connections")
  - multiple requests at beginning of session, several responses in same order ("pipelining")
  - semantic transparency not always achievable: disconnected, performance, availability, re-transmission, most up-to-date version...
  - several more tags and options for controlling caching (public/private, max-age, no-cache etc.)
  - relaxing of transparency on app. request or with warning to user
  - encoding/compression mechanism, integrity check, security of proxies, authentication, authorization...

- Cookies: well..., stateful sessions, not really integrated...

System support for WWW in a mobile world I

- Enhanced browsers
  - Pre-fetching, caching, off-line use
  - e.g. Internet Explorer

- Additional, accompanying application
  - Pre-fetching, caching, off-line use
  - e.g. original WebHacker
System support for WWW in a mobile world II

- **Client Proxy**
  - Pre-fetching, caching, off-line use
  - e.g., Caubweb, TeleWeb, Weblicator, WebWhacker, WebEx, WebMirror, ...

- **Network Proxy**
  - Adaptive content transformation for bad connections, pre-fetching, caching
  - e.g., TranSend, Digestor

System support for WWW in a mobile world III

- **Client and network proxy**
  - Combination of benefits, plus simplified protocols
  - e.g., MobiScape, WebExpress

- **Special network subsystem**
  - Adaptive content transformation for bad connections, pre-fetching, caching
  - e.g., Mowgli

WAP - Wireless Application Protocol

- **Goals**
  - Deliver Internet content and enhanced services to mobile devices and users (mobile phones, PDAs)
  - Independence from wireless network standards
  - Open for everyone to participate, protocol specifications will be proposed to standardization bodies
  - Applications should scale well beyond current transport media and device types and should also be applicable to future developments

- **Platforms**
  - e.g., GSM (900, 1800, 1900), CDMA IS-95, TDMA IS-136, 3rd generation systems (IMT-2000, UMTS, W-CDMA, cdma2000 1x EV-DO, ...)

- **Forum**
  - Was: WAP Forum, co-founded by Ericsson, Motorola, Nokia, Unwired Planet, further information www.wapforum.org
  - Now: Open Mobile Alliance www.openmobilealliance.org
  - (Open Mobile Architecture + WAP Forum + SyncML + ...)
WAP Architecture

WAP Gateway
WML Encoder
WMLScript Compiler
Protocol Adapter

Web Server
CGI Scripts etc.

WAP Gateway
WML Encoder
WMLScript Compiler
Protocol Adapter

WAP Application Server

WAP 1.x - reference model and protocols

Internet
- HTML, Java
- HTTP
- SSL/TLS
- TCP/IP, UDP/IP, media

WAP
- Application Layer (WAE)
- Session Layer (WSP)
- Transaction Layer (WTP)
- Security Layer (WTLS)

additional services and applications

WAE comprises WML (Wireless Markup Language), WML Script, WTAI etc.
Why not HTML/HTTP over wireless?

WDP - Wireless Datagram Protocol

- Protocol of the transport layer within the WAP architecture
  - uses the transport mechanisms of different network technologies
  - offers a common interface for higher layer protocols
  - allows for transparent communication using different transport technologies (GSM [SMS, CSD, USSD, GPRS, ...], IS-136, TETRA, DECT, PHS, IS-95, ...)

- Goals of WDP
  - create a worldwide interoperable transport system with the help of WDP adapted to the different underlying technologies
  - transmission services such as SMS, GPRS in GSM might change, new services can replace the old ones

- Additionally, WCMP (wireless Control Message Protocol) is used for control/error report (similar to ICMP in the TCP/IP protocol suite)
WDP - Service Primitives

Usage of WDP

WTLS - Wireless Transport Layer Security

Goals

- data integrity
  - prevention of changes in data

- privacy
  - prevention of tapping

- authentication
  - creation of authenticated relations between a mobile device and a server

- protection against denial-of-service attacks
  - protection against repetition of data and unverified data

WTLS

- is based on the TLS (Transport Layer Security) protocol (former SSL, Secure Sockets Layer)
- optimized for low-bandwidth communication channels
Secure session, full handshake

SEC-Create.req
(SA, SP, DA, DP, KES, CS, CM)

SEC-Create.ind
(SA, SP, DA, DP, KES, CS, CM)

SEC-Create.cnf
(SNM, KR, SID, KES', CS', CM')

SEC-Create.res
(SNM, KR, SID, KES', CS', CM')

SEC-Exchange.req

SEC-Exchange.ind

SEC-Exchange.cnf

SEC-Exchange.res

SEC-Commit.req

SEC-Commit.ind

SEC-Commit.cnf

SEC-Unidata - transferring datagrams

SEC-Unidata.req
(SA, SP, DA, DP, UD)

SEC-Unidata.ind
(SA, SP, DA, DP, UD)

WTP - Wireless Transaction Protocol

- Goals
  - different transaction services, offloads applications
  - application can select reliability, efficiency
  - support of different communication scenarios
    - class 0: unreliable message transfer
    - class 1: reliable message transfer without result message
    - class 2: reliable message transfer with exactly one reliable result message
  - low memory requirements, good for simple devices (< 10kbyte)
  - efficient for wireless transmission
    - segmentation/reassembly
    - selective retransmission
    - header compression
    - no connection setup required
Details of WTP I
- Support of different communication scenarios
  - Class 0: unreliable message transfer
    - Example: push service
  - Class 1: reliable request
    - An invoke message is not followed by a result message
    - Example: reliable push service
  - Class 2: reliable request/response
    - An invoke message is followed by exactly one result message
    - With and without ACK
    - Example: typical web browsing
- No explicit connection setup or release is available
- Services for higher layers are called events

Details of WTP II
- Used Mechanisms
  - Reliability
    - Unique transaction identifiers (TID)
    - Acknowledgements
    - Selective retransmission
    - Duplicate removal
  - Optional: concatenation & separation of messages
  - Optional: segmentation & reassembly of messages
  - Asynchronous transactions
  - Transaction abort, error handling
  - Optimized connection setup (includes data transmission)

WTP Class 0 transaction

WTP invoke sequence:

1. Initiator sends TR-Invoke.req
2. Initiator sends TR-Invoke.ind
3. Initiator sends TR-Invoke.req
4. Initiator sends TR-Invoke.ind
5. Initiator sends TR-Invoke.req
6. Initiator sends TR-Invoke.ind

Diagram:

- Initiator sends TR-Invoke.req (SA, SP, DA, DP, A, UD, C=0, H)
- Initiator sends TR-Invoke.ind (SA, SP, DA, DP, A, UD, C=1, H)
- Initiator sends TR-Invoke.req (SA, SP, DA, DP, A, UD, C=0, H)
- Initiator sends TR-Invoke.ind (SA, SP, DA, DP, A, UD, C=1, H)
WTP Class 1 transaction, no user ack & user ack

WTP Class 2 transaction, no user ack

WTP Class 2 transaction, user ack
WTP Class 2 transaction, hold on, no user ack