CSE 4215/5431:
Mobile Communications

Winter 2013

Suprakash Datta

datta@cse.yorku.ca

Office: CSEB 3043

Phone: 416-736-2100 ext 77875

Course page: http://www.cse.yorku.ca/course/4215

Some slides are adapted from the book website

Next

Support for mobility

- File systems
- Data bases
- WWW and Mobility

Mobile, bearable multimedia equipment

File systems - Motivation

- Goal
  - efficient and transparent access to shared files within a mobile environment while maintaining data consistency
- Problems
  - limited resources of mobile computers (memory, CPU, ...)
  - low bandwidth, variable bandwidth, temporary disconnection
  - high heterogeneity of hardware and software components (no standard PC architecture)
  - wireless network resources and mobile computer are not very reliable
  - standard file systems (e.g., NFS, network file system) are very inefficient, almost unusable
- Solutions
  - replication of data (copying, cloning, caching)
  - data collection in advance (hoarding, pre-fetching)

File systems - consistency problems

- THE big problem of distributed, loosely coupled systems
  - are all views on data the same?
  - how and when should changes be propagated to what users?
- Weak consistency
  - many algorithms offering strong consistency (e.g., via atomic updates) cannot be used in mobile environments
  - invalidation of data located in caches through a server is very problematic if the mobile computer is currently not connected to the network
  - occasional inconsistencies have to be tolerated, but conflict resolution strategies must be applied afterwards to reach consistency again
- Conflict detection
  - content independent: version numbering, time-stamps
  - content dependent: dependency graphs

File systems for limited connectivity I

- Symmetry
  - Client/Server or Peer-to-Peer relations
  - support in the fixed network and/or mobile computers
  - one file system or several file systems
  - one namespace for files or several namespaces
- Transparency
  - hide the mobility support, applications on mobile computers should not notice the mobility
  - user should not notice additional mechanisms needed
- Consistency model
  - optimistic or pessimistic
- Caching and Pre-fetching
  - single files, directories, subtrees, partitions, ...
  - permanent or only at certain points in time

File systems for limited connectivity II

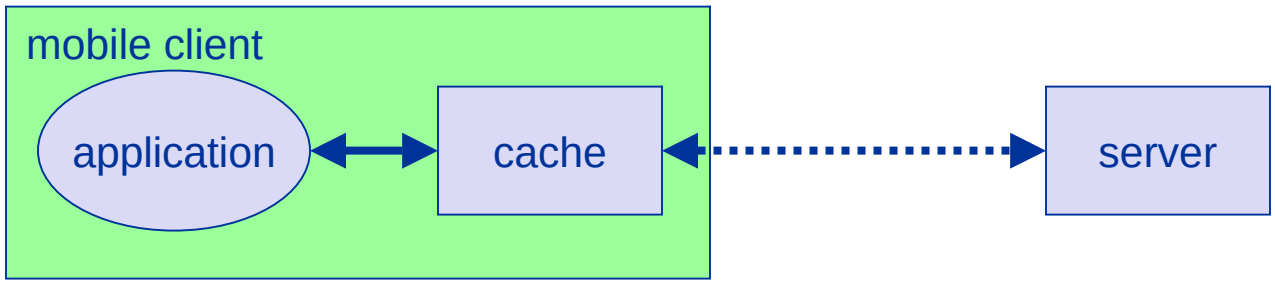
- Data management
  - management of buffered data and copies of data
  - request for updates, validity of data
  - detection of changes in data
- Conflict solving
  - application specific or general
  - errors

- Several early experimental systems exist (late 80s)
  - Coda (Carnegie Mellon University), Little Work (University of Michigan), Ficus (UCLA) etc.

- Many systems use ideas from distributed file systems such as, e.g., AFS (Andrew File System)

File systems - Coda I

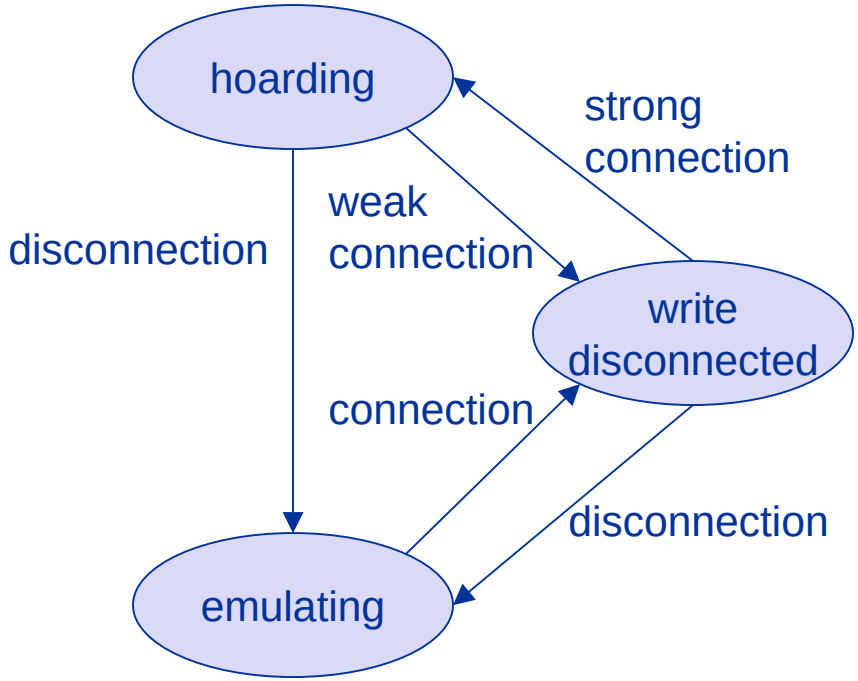
- Application transparent extensions of client and server
  - changes in the cache manager of a client
  - applications use cache replicates of files
  - extensive, transparent collection of data in advance for possible future use („Hoarding“)
- Consistency
  - system keeps a record of changes in files and compares files after reconnection
  - if different users have changed the same file a manual reintegration of the file into the system is necessary
  - optimistic approach, coarse grained (file size)



File systems - Coda II

- Hoarding
  - user can pre-determine a file list with priorities
  - contents of the cache determined by the list and LRU strategy (Last Recently Used)
  - explicit pre-fetching possible
  - periodic updating
- Comparison of files
  - asynchronous, background
  - system weighs speed of updating against minimization of network traffic
- Cache misses
  - modeling of user patience: how long can a user wait for data without an error message?
  - function of file size and bandwidth

- States of a client



File systems - Little Work

- Only changes in the cache manager of the client
- Connection modes and use

	Connected	**Partially Connected**	**Fetch only**	**Disconnected**
Method	normal	delayed write to the server	optimistic replication of files	abort at cache miss
Network requirements	continuous high bandwidth	continuous bandwidth	connection on demand	none
Application	office, WLAN	packet radio	cellular systems (e.g., GSM) with costs per call	independent

File systems - further examples

- Mazer/Tardo
  - file synchronization layer between application and local file system
  - caching of complete subdirectories from the server
  - “Redirector” responses to requests locally if necessary, via the network if possible
  - periodic consistency checks with bi-directional updating
- Ficus
  - not a client/server approach
  - optimistic approach based on replicates, detection of write conflicts, conflict resolution
  - use of „gossip“ protocols: a mobile computer does not necessarily need to have direct connection to a server, with the help of other mobile computers updates can be propagated through the network
- MIo-NFS (Mobile Integration of NFS)
  - NFS extension, pessimistic approach, only token holder can write
  - connected/loosely connected/disconnected

Database systems in mobile environments

- Request processing
  - power conserving, location dependent, cost efficient
  - example: find the fastest way to a hospital
- Replication management
  - similar to file systems
- Location management
  - tracking of mobile users to provide replicated or location dependent data in time at the right place (minimize access delays)
  - example: with the help of the HLR (Home Location Register) in GSM a mobile user can find a local towing service
- Transaction processing
  - “mobile” transactions can not necessarily rely on the same models as transactions over fixed networks (ACID: atomicity, consistency, isolation, durability)
  - therefore models for “weak” transaction

World Wide Web and mobility

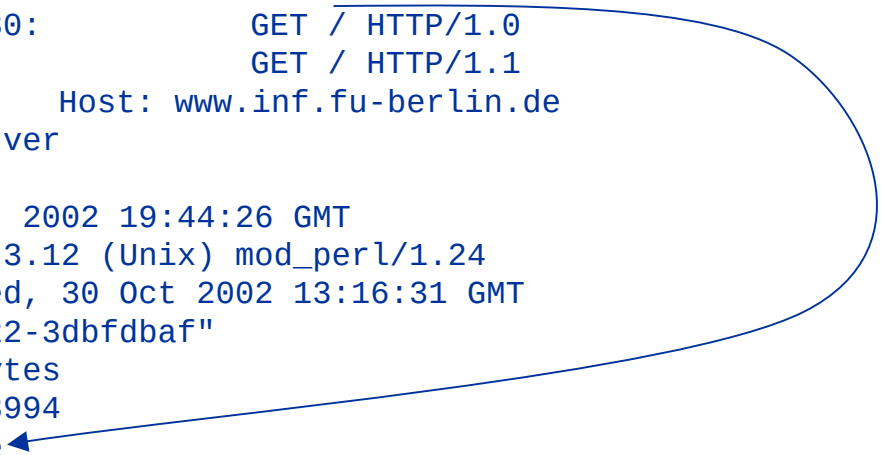
- Protocol (HTTP, Hypertext Transfer Protocol) and language (HTML, Hypertext Markup Language) of the Web have not been designed for mobile applications and mobile devices, thus creating many problems!
- Typical transfer sizes
  - HTTP request: 100-350 byte
  - responses avg. <10 kbyte, header 160 byte, GIF 4.1kByte, JPEG 12.8 kbyte, HTML 5.6 kbyte
  - but also many large files that cannot be ignored
- The Web is no 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
Last-Modified: Wed, 30 Oct 2002 13:16:31 GMT
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 f&uuml;r 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();">
...
```



non persistent

HTTP 1.0 (old) 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-handshake) compared with the content, e.g., of a GET request
    - slow-start problematic
  - DNS lookup by client causes additional traffic

HTTP 1.0 (old) 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
    - how to use SSL/TLS together with proxies?
  - today: many user customized pages, dynamically generated on request via CGI, ASP, ...
- POSTing (i.e., sending to a server)
  - can typically not be buffered, very problematic if currently disconnected
- Many unsolved problems!

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)
- Additional “features”
  - animated GIF, Java AWT, 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 toward WWW for mobile devices

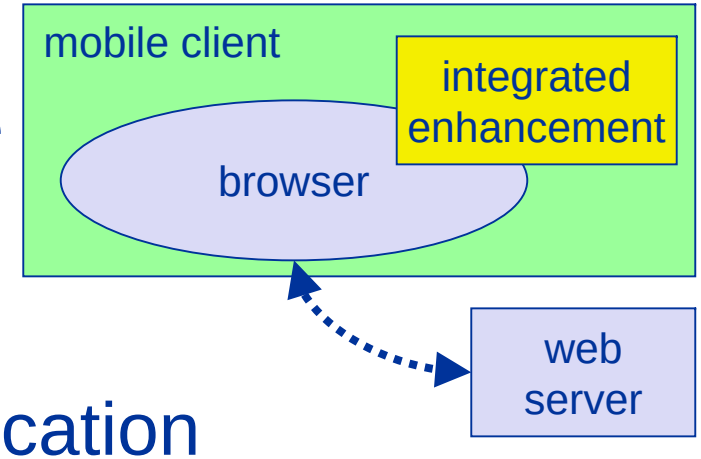
- Application gateways, enhanced servers
  - simple clients, pre-calculations in the fixed network
  - compression, filtering, content extraction
  - automatic adaptation to network characteristics
- Examples
  - picture scaling, color reduction, transformation of the document format (e.g., PS to TXT)
  - detail studies, clipping, zoom
  - 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?

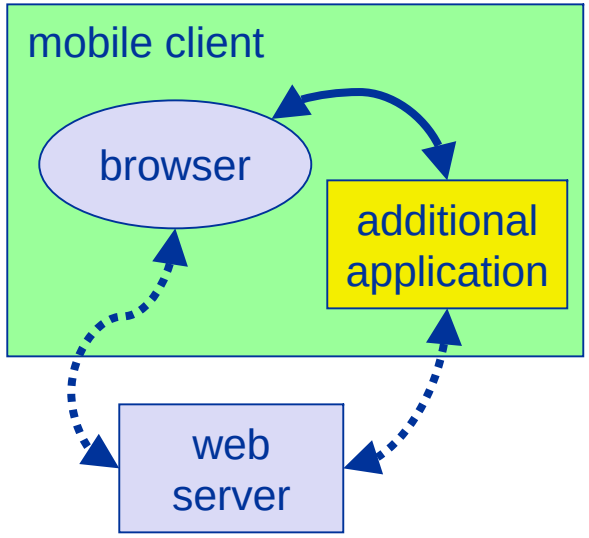
- Push technology
  - real pushing, not a client pull needed, channels etc.
- HTTP/1.1
  - client/server use the same connection for several request/response transactions
  - multiple requests at beginning of session, several responses in same order
  - enhanced caching of responses (useful if equivalent responses!)
  - semantic transparency not always achievable: disconnected, performance, availability -> 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

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

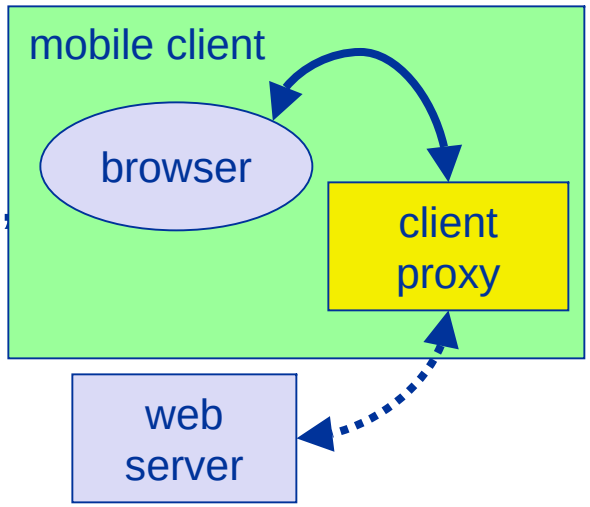


- Additional, accompanying application
  - Pre-fetching, caching, off-line use
  - e.g. original WebWhacker

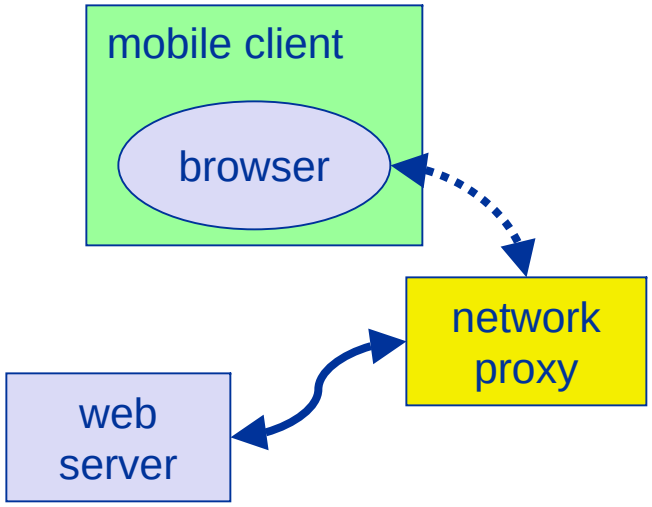


System support for www in a mobile world

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

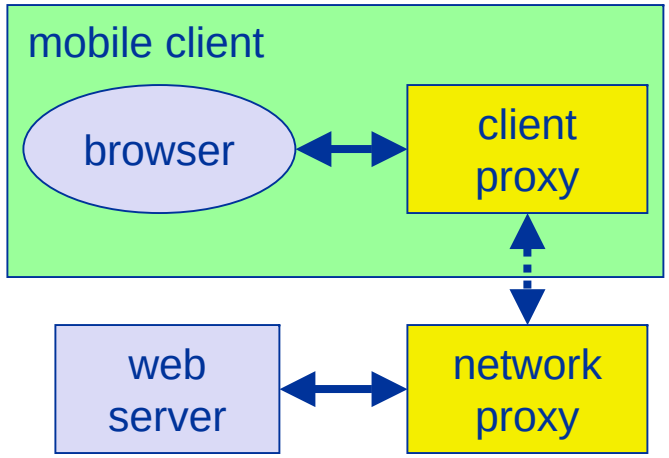


- Network Proxy
  - adaptive content transformation for bad connections, pre-fetching, caching
  - e.g., TranSend, Digestor

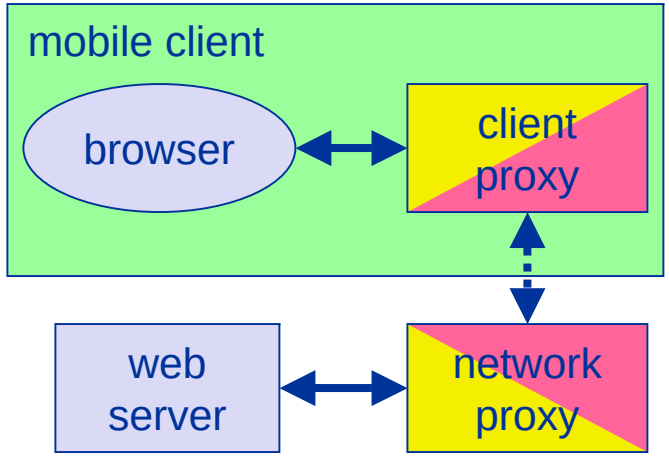


System support for www in a mobile world

- 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



- Additional many proprietary server extensions possible
  - “channels”, content negotiation, ...