Mobile Applications

Emmanuel Agu
MobiDesk

- **Mobile Virtual Desktop Computing**
- **Goal:** Virtualize display, OS, networking
- *No modification to application code, OS, networking*
- **Example:** mobile device runs ASP application

**Benefits:**

- *Faulty hosts, server upgrade:* Easily migrate applications
- *Run sensitive applications on most secure servers*
- *Mobile users:* need basic internet access, no pressure to upgrade, can connect and compute from anywhere
- *ASP:* offer many applications, many users, amortize costs
- *Subscription* service for all types of users
Overview of System Architecture

- Proxy-based server cluster:
  - proxy, servers, network storage, LAN, clients

Figure 1: MobiDesk Architecture
Virtualization?

session environment decoupled from underlying physical infrastructure

PC

user session

OS Display Net

MobiDesk

user session

virtualization + translation

OS Display Net
System Overview

- Clients are simple input-output devices
- Users interact with MobiDesk sessions through session viewer
- Session viewer relays user input and session output through secure channel
- Each user assigned username, password
- Users unaware of each other
System Overview

• User log out, session continues, can reconnect
• Dynamically relocate sessions to balance load
• Sessions can be checkpointed, migrated anytime
• *Session cookies* created for each session, passed between servers during migration
• Dest. server informs proxy of application location
Display Virtualization

- Virtual display runs local to application
- Actual display on user’s device
- Create virtual display driver
- Virtual display driver
  - intercepts drawing commands from user’s application
  - Translates commands client-server into display protocol
  - Tries to minimize latency for interactive applications
  - Anticipates higher future bandwidth (Korea 100mbps)
Display Virtualization

- Applications
- Window system
- Virtual device driver
- Framebuffer

Display updates
Input events
Display Virtualization

- Security: client-server communication encrypted
- Client hardware support used for speedup
  - YUV video format used because supported by hardware
  - Color space conversion, scaling done at client
- Cursor state maintained at client
- Automatic resizing for variable client screens
- Push @ server: update client once data available
- Shortest message first: IM, clicking (short packets) need to be sent first
- Thin client, but use client resources if available
Display Virtualization

- Client-server display protocol commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAW</td>
<td>Display raw pixel data at a given location and size</td>
</tr>
<tr>
<td>COPY</td>
<td>Copy frame buffer area to specified coordinates</td>
</tr>
<tr>
<td>SFILL</td>
<td>Fill an area with a given pixel color value</td>
</tr>
<tr>
<td>BITMAP</td>
<td>Fill a region using a bitmap image</td>
</tr>
<tr>
<td>PFILL</td>
<td>Tile a pixmap rectangle in a given region</td>
</tr>
</tbody>
</table>

Table 1: MobiDesk Protocol Display Commands
OS Virtualization

• Private namespace for each hosted client session
• Session have host-independent view of OS resources (PID/GID, memory, filesystem, devices)
• MobiDesk virtualizes sessions, VMWare virtualizes entire OS
• Virtualizing session: migration is possible even if OS being upgraded
How OS Virtualization works

• Session virtualization
  – Process resource gets virtual name mapped to OS physical name
  – Process creates resource, OS-assigned name is caught
  – Private virtual name is created and returned instead
  – Anytime OS passes virtual name, caught and replaced with physical OS name
OS Virtualization

- Session migration
  - Possible between machines with common CPU arch.
  - Checkpoint-restart mechanism
  - Save process image, digitally signed (integrity)
  - Saves high-level semantic, not low-level kernel information
  - Info: \{virtual source pid, source fd, virtual dest. pid, dest. Pid\}

- Processes within session communicate with IPC
- Processes outside session, do RPC
Network Virtualization

- Persistent network connection for mobile clients
- All connections through proxy
- Virtual private namespace for IP address, port no.
  - Virtual identifiers remain constant
  - Translation to actual IP address, port no. changes
- Use DHCP if client, server on same subnet
  - Range of IP address for servers, other range for MobiDesk
  - Proxy maps mobiDesk IP to real server as alias
- Mobile client: remap!
  - ARP used for link-layer resolution
Network Virtualization

- DHCP breaks down if client strays into network with new IP address
- Solution: use two IP addresses (physical, virtual)
Testbed

- Linux environment
- XFree86 display module
- Xlib, Java client

Figure 4: Experimental Testbed
Results

• Good results
  – Good web page latency
  – Good video quality
  – Reasonable Overhead (<10%)

• Okay results
  – Web data transferred (iBench benchmark)
  – Video transfer
  – Session migration worked
Web Page Latency

Figure 10: Average Per Web Page latency
Video Playback Performance

- Video quality: playback time and frames displayed at the client

  Example: 50% video quality
  - Twice as long to play the video, or
  - Half of the frames were not displayed
Video Quality

Figure 12: Video Quality
Session Migration

• Dynamic migration, checkpointing, worked
Conclusions

- MobiDesk virtualization concept was validated
- Worked well with unmodified OS, application, networking code
- Ubiquitous access to applications
Wireless Applications

• Wireless/mobile applications:
  – Mobile filesystems
  – Wireless messaging: SMS, etc
Mobile, bearable multimedia equipment ...
File systems - Motivation

• **Goal**
  – efficient and transparent mobile access to shared files while maintaining data consistency

• **Problems**
  – Mobile terminals can frequently disconnect
  – standard file systems (e.g., NFS, network file system) see disconnection as an error

• **Solutions**
  – replication of data (copying, cloning, caching)
  – data prefetching, hoarding, pre-fetching
File systems - consistency problems

• Problem of distributed, loosely coupled systems
  – are all views on data the same?
  – Write issues: how should changes be propagated to users?

• Weak consistency
  – strong consistency (e.g., via atomic updates) cannot be used in mobile environments
  – invalidation of cached data through a server is problematic if the mobile computer is disconnected
  – occasional inconsistencies have to be tolerated, but conflict resolution strategies needed to achieve consistency again

• Conflict detection
  – content independent: version numbering, time-stamps
  – content dependent: dependency graphs
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 change same file, manual reintegration is necessary (Note: 0.3% concurrent writes)
  – optimistic approach, coarse grained (file size)
File systems - Coda II

- Good connection:
  - Hoarding
- Total disconnection:
  - Emulating state
- Weak connection:
  - Write disconnected
  - No hoarding
  - Decide if to fetch file on cache miss based on type

Client states:

- hoarding
- emulating
- write disconnected

Connection types:
- strong
- weak
- disconnection
File systems - Little Work

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

<table>
<thead>
<tr>
<th></th>
<th>Connected</th>
<th>Partially Connected (traffic cost)</th>
<th>Fetch only (E.g. GSM /call costs)</th>
<th>Disconnected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>normal</td>
<td>delayed write to the server</td>
<td>optimistic replication of files</td>
<td>abort at cache miss</td>
</tr>
<tr>
<td>Network requirements</td>
<td>continuous high bandwidth</td>
<td>continuous bandwidth</td>
<td>connection on demand</td>
<td>none</td>
</tr>
<tr>
<td>Application</td>
<td>office, WLAN</td>
<td>packet radio</td>
<td>cellular systems (e.g., GSM) with costs per call</td>
<td>independent</td>
</tr>
</tbody>
</table>
File systems - further examples

• 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, token holder can write
  – connected/loosely connected/disconnected
Wireless Messaging

• Quick word on wireless messaging:
  – Email is still killer application on the Internet
  – Instant messaging also very huge growth
  – Messaging available on certain wireless phones
• Short Messaging Service (SMS) was part of original GSM 2G cellular network in Europe
• Most 2G and 2.5G phones can send some form of SMS
• SMS is sometimes hooked up to AOL, MSN, Yahoo messenger
• Popularity of SMS led to other messaging standards:
  – CBS (broadcast messages)
  – USSD (connection-oriented, can reply immediately)
  – Enhanced or Smart messaging (fonts, concatenate msgs, etc)
  – Multimedia messaging (graphics, multimedia)
System support for mobile WWW I

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

- Additional application
  - Pre-fetching, caching, off-line use
  - e.g. original WebWhacker
System support for mobile WWW 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 mobile WWW 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

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

- Ref: *Computer Networks by Tanenbaum (4th edition)*
- Today’s web model
  - You click on a page, HTML page and linked stuff retrieved
  - Page is retrieved in network packets (packet switched)
  - Success of web made people want to access it wirelessly
- Wireless Application Protocol (WAP) 1.0
  - Application protocol stack for wireless web
  - Standard proposed by consortium which included Nokia, Ericsson, Motorola, and Phone.com (prev. Unwired planet)
  - WAP device may be mobile phone, PDA, notebook, etc
  - WAP optimized for mobile device (low CPU, memory, screen), low-bandwidth wireless links
WAP 1.0

- **WAP 1.0**
  - Brute force approach
    - Make phone call to web gateway
    - Send URL to gateway
    - If available, gateway returns page
  - Issues:
    - Connection-oriented (circuit-switched, per-minute billing), charged while reading web page
    - WAP pages written in Wireless Markup Language (WML) (major drawback: No HTML)
    - WML is XML-based
    - Sometimes a WAP filter (server) can automatically convert HTML pages to WML
  - Result: failed, but laid groundwork for iMode and WAP 2.0
## WAP Protocol Stack

- Six layers (including actual wireless network)
- WDP is datagram protocol, similar to UDP
- WTLS is security layer, subset of Secure Socket Layer by Netscape
- WTP is similar to TCP, concerned with requests and responses
- WSP is similar to HTTP/1.1
- WAE is microbrowser

<table>
<thead>
<tr>
<th>Protocol Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wireless Application Environment (WAE)</td>
<td></td>
</tr>
<tr>
<td>Wireless Session Protocol (WSP)</td>
<td></td>
</tr>
<tr>
<td>Wireless Transaction Protocol (WTP)</td>
<td></td>
</tr>
<tr>
<td>Wireless Transport Layer Security Protocol (WTLS)</td>
<td></td>
</tr>
<tr>
<td>Wireless Datagram Protocol (WDP)</td>
<td></td>
</tr>
<tr>
<td>Bearer Layer (GSM, CDMA D-AMPS, GPRS, etc)</td>
<td></td>
</tr>
</tbody>
</table>
I-Mode

- Sometimes in telecom, single organization or person beats consortium. E.g. Jon Postel developed RFCs for TCP, SMTP, etc.
- In parallel to WAP effort, Japanese woman Mari Matsunaga created a different approach called I-Mode (Information Mode).
- Mari convinced Japanese telco monopoly, NTT DoCoMo to deploy service.
- I-Mode subscription exploded!!
- 35 million Japanese subscribers in 3 years, access to 40,000 I-Mode pages.
- Major financial success!
- Interesting case study: features, why it succeeded?
I-Mode

• To make I-Mode work, 3 new components:
  – New transmission system (partnership with Fujitsu)
  – New handset (partnered with NEC, Matsushita)
  – New web page language (cHTML)

• Transmission system - 2 separate networks:
  – Voice mode:
    • old 2G digital phone network, PDC
    • (circuit-switched),
    • billed per connected minute
  – I-Mode:
    • New packet-switched network for I-Mode, always on
    • Internet connection, users unaware of this!
    • No connection charge, billed per packet sent
    • Uses CDMA, 128-byte packets at 9600 bps
  – Both networks cannot be used simultaneously
I-Mode

• I-Mode handsets:
  – Enhanced features with CPU power of PC in 1995
  – Small screen
  – IP-capable communications

• Handset specifications
  – 100 MHz CPU
  – Memory: Several MB flash memory, 1MB RAM
  – Dimensions: smaller than pack of cigarettes, 70 grams
  – Screen:
    • Resolution: min. 72 x 94 pixels, 120 x 160 high end
    • Color: 256 colors initially, good for line drawings, cartoons, no photographs. New: 65,000 colors
  – Navigation: no mouse, use arrow keys, “i” key takes you to I-mode services menu
i-mode examples I
I-Mode

- I-Mode handsets:
  - When user hits “i” key on handset, user presented with list of Categories: email, news, weather, sports, etc (a portal)
  - over 1000 “services” in about 20 categories
  - Lots of services targetted at teenagers, young people
  - Each service is I-Mode website run by independent company
  - May type in service URL directly also
  - Users subscribe to services ($1-$2 per service)
  - > 1,000,000 subscriber makes service official
  - Official services billed through phone bill
  - 1500 official services, 39,000 unofficial circa 2001
I-Mode

- **I-Mode handsets:**
  - Most popular application is email: limit of 500 bytes (SMS on GSM limit is 160 bytes)
  - I-Mode phone number doubles as email address (e.g. 0345671234@docomo.co.jp)
  - Rich in graphics content, Japanese have high visual sensibility
  - Invented new cute pictograms like smileys called emoji
  - US company, Funmail has patented text-to-graphics. E.g. word Hawaii in email may be converted to animated cartoon image of “beach with swaying palm trees”
  - Funmail is multi-platform technology:
    - cell phones receive animations scaled for power, screen size.
    - Desktops receive full-blown animation
I-Mode

- I-Mode is massive success in Japan because:
  - Few people own PCs
  - Local phone access is expensive
  - Lots of time spent commuting
- Different circumstances for US and Europe
- I-Mode structure and operation:
  - Handsets speak Lightweight Transport Protocol (LTP) over wireless link to protocol conversion gateway
  - Gateway converts request to TCP request
  - Gateway has fiber-optic connection to I-Mode server
  - I-Mode server caches most pages for performance
I-Mode

- I-Mode protocol stack:

  - User Interaction module
  - Plug-ins
  - cHTML interpreter
  - Java
  - Simple window manager
  - Network communication (LTP)
  - Real-time operating system

- I-Mode pages programmed in cHTML
- Java functionality based on J2ME (Java 2 Platform Micro Edition) based on the Kilobyte Virtual Machine (KVM)
- Maximum of 5 applets can be stored at a time
I-Mode

- **cHTML**
  - Developed by Access, embedded software maker
  - based on HTTP 1.0, with omissions and extensions
  - Most HTML tags allowed. E.g. `<body>`, `<ul>`, `<br>`, etc
  - New tag to dial phone number, phoneto
  - E.g. phoneto on a restaurant’s page lets you dial number
  - HTML-based: can view I-Mode pages on regular browser

**I-Mode Browser:**
- Limited
- Allows plug-ins and helper applications e.g. JVM
- No Javascript support, frames, background colors/images, JPEG (takes too long)

**I-Mode Server-side:**
- Full-blown computer, all bells and whistles
- Supports CGI, Perl, PHP, JSP, ASP, most web standards
WAP 2.0

- Goal: fix WAP 1.0 shortcomings
- Features:
  - Push model as well as pull
  - Integrated telephony (voice and data) into applications
  - Multimedia messaging
  - Include 264 pictograms (emoji)
  - Interface to storage device (e.g. flash memory)
  - Support for browser plug-ins (also new scripting language, WMLScript)
WAP 2.0

• New protocol stack based on TCP and HTTP/1.1
• Modified TCP (compatible with original)
  – Fixed 64KB window
  – No slow start
  – Maximum 1500-byte packet
  – Slightly different transmission algorithm
• WAP 2.0 supports new and old (WAP 1.0) protocol stack
WAP 2.0

- WAP 2.0 supports XHTML basic
- NTT DoCoMo has agreed to support XHTML so that pages will be widely compatible
- Hopefully, this will end format wars
- XHTML targetted at low end devices (mobile phones, TVs, PDAs, vending machines, pagers, watches, etc)
- Thus, no style sheets, scripts or frames
- WAP 2.0 speed 384 kbps
- WAP threat:
  - 802.11b (11Mbps) and 802.11g (54Mbps) can download regular web pages, becoming available in coffee shops
  - People will prefer 802.11 where available
- Hybrid solution: dual mode devices that use 802.11 where available and WAP otherwise
Projects

- Term long project (half of course grade)
- Preferably work in groups
- You should work on things you enjoy, good at
- Need to decide:
  - Top 3 areas you may like to explore (deadline Feb. 7)
  - Partner(s)?
  - Your strengths/weaknesses are
  - Nature of research you like doing
    - Mathematical
    - Algorithmic
    - Experimental/measurement
    - Simulation
    - System design and development
Projects

- 4 Project deadlines:

<table>
<thead>
<tr>
<th>Description</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form team, decide proj. area:</td>
<td>February 10</td>
</tr>
<tr>
<td>Propose project:</td>
<td>March 14</td>
</tr>
<tr>
<td>Mid-project update:</td>
<td>April 4</td>
</tr>
<tr>
<td>Present results</td>
<td>April 25</td>
</tr>
</tbody>
</table>

- Note: **March 7: no class (term break)**
Presentations

• Approx. 50-minute talk based on selected paper
• Make sure you understand the paper
• Select:
  – Aspects to present/omit
  – Supplemental material to add to improve understanding
• Rough talk outline (of research paper):
  – Introduce problem/give overview
  – Explain main proposed solutions
  – Other improvements
  – Future work
Presentations

• This powerpoint template is on website. Please use for uniformity!!
• Note: send me your powerpoint slides latest noon on the day of your talk, so that I can put it on website
• If you are unsure of how to use your 50 minutes, you can ask me. E.g. if paper looks long
• You can send me outlines, rough drafts of slides, etc.