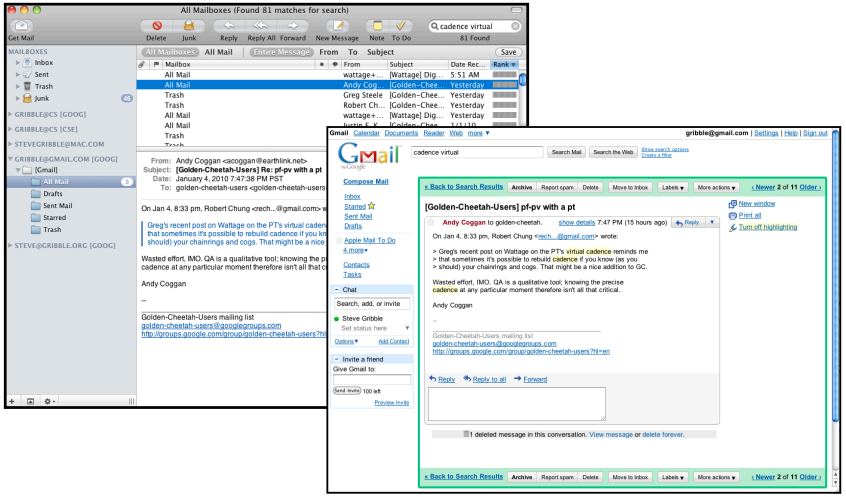
CSE599w winter '10
operating systems and the web

***introduction***

Department of Computer Science & Engineering
University of Washington



# Clearly something is happening...






# and it's picking up steam...

| PC | Web |
|---|---|
| MS office | Office Web, Google docs, Zimbra, ... |
| Mail | Gmail, hotmail... |
| iTunes | Pandora, Hulu, ... |
| Skype, iChat | WebEx, Google voice, twitter, ... |
| Quicken | mint.com, ... |
| iPhoto | flickr, photobucket, ... |
| Eclipse | bespin, heroku, ... |



# but the path forward isn't as clear.

**Native Client: A Sandbox for Portable, Untrusted x86 Native Code**

**ChromeOS**

**W3C HTML5**

Powered by ADOBE® AIR®

palm webOS

VMware View™
TRANSFORM DESKTOP MANAGEMENT, ENERGIZE USERS

Microsoft® Silverlight

# Browsers from 10,000’



# Browsers: their inception

Netscape 1-2 (1994-1996)

- first to “level the playing field” among OSs
- HTML, cookies, SSL
- grew to 90% market share

Web page == static document

- no security concerns besides server access control
- major debate about markup vs. presentation
- small, stable “A”PI



# “The Internet Tidal Wave”

Bill Gates memo to Microsoft executives, May 26, 1995

> “ A new competitor ‘born’ on the Internet is Netscape. Their browser is dominant, with 70% usage share, allowing them to determine which network extensions will catch on. They are pursuing a multi-platform strategy where they move the key API into the client to commoditize the underlying OS.
>
> ...
>
> One scary possibility being discussed by Internet fans is whether they should get together and create something far less expensive than a PC which is powerful enough for Web browsing.”



# The first browser wars

Netscape 3, IE 3-5 (1996-1999)

- Microsoft wants in, and we’re off to the feature races
  - ActiveX, JavaScript, Java, BHOs, inline multimedia, i18n, XML, CSS/HTML, Ajax (XmlHttpRequest)
- rapid innovation but lots of friction
  - divergence and incompatibility between Netscape & IE
  - security/privacy issues galore
- IE wins the war by bundling and cutting distribution deals (“cut off air supply”)

Web page == document with a little teensy bit of code

- but JavaScript is largely considered evil
- the Web is still a content publishing and distribution platform

# Browsers: the doldrum years

Netscape capitulates (1999-2004)

- Navigator is open sourced as Mozilla, suffers feature creep, Firefox is born (2003)
- without competition, IE stagnates in version 6
  - IE6 released in 2001, IE7 not out until 2006!!
  - a period of very slow innovation in browser technology

But, some harbingers of what is to come

- a few Web apps steal appreciable market share from desktop apps
  - e.g., hotmail, Google maps
- JavaScript and AJAX are truly embraced and Flash becomes pervasive
  - "rich Internet apps" (RIA)
  - **Web page ==> Web program, browser --> OS**

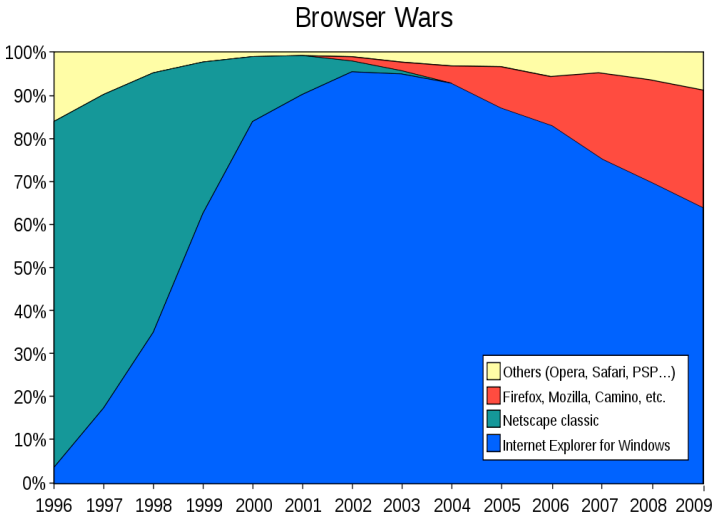


# The second browser wars

Renewed biodiversity and competition (2006 -- ?)

- a bloom of browsers from tough competitors (Safari, Chrome, Firefox)
- Google emerges as the Microsoft of the Web
  - "search, ads and apps"
  - search and ads are their cash cow; apps, browsers as delivery vehicle
  - "donates" browser technology and apps to drive more Web usage
- Microsoft starts sweating again (Gates' "Internet Software Services" memo)
  - the resulting competition revitalizes Web browser technologies
    - process oriented browsers, strong sandboxing, effective Flash video, HTML5, fast (and secure?) JavaScript, ...
    - the mobile Web, but for real (not WAP, thank god!)






Chrome: 4.6%
Safari: 4.5%
Opera: 2.4%

Firefox: 25%

IE: 63%

from wikipedia [http://en.wikipedia.org/wiki/File:Layout_engine_usage_share.svg]



# Why is the Web so attractive?

The Web is everywhere for everyone

- your software and data are not tethered to any computer [disposable]
- to first order, every Web app runs on any OS+browser stack [portable]

Software distribution is no longer an issue

- app vendors deliver users a new program on every mouse click
- vendors don't need to deal with configuration complexity problem

The Web encourages connecting apps, people, data

- mashups, Web REST APIs, html/JavaScript/image/media embeds
- multi-tenant Web services and the network effect
- services backed by massive data sets, compute farms

# But why are browsers so lousy?

A stone-aged development platform

- a single slow & scruffy language is supported
- a program is a single threaded event handler without any offline execution
- its “system call interface” (DOM) lacks many conveniences
  - no stable storage
  - no access to local devices (cameras, mikes, GPUs, ...)
  - no real IPC
- programs are handcuffed by an esoteric security model (SOP)



# Counterpoint: desktop OSs



# Massive strengths

Rich, stable, powerful APIs and process model

- 20 years of Win32 and POSIX, 10 years of .NET
- convenient access to the bounty of Moore’s law
  - multithreaded / multiprocess, indexed local file systems, GPU access, multimedia libraries, direct TCP access
- develop in any language you like, execute with native speed, portable across hardware configurations
- offline support!!
- in theory, the user can fully control their environment and data



# But also massive weaknesses

OSs still seem to have poor application isolation

- installing an app can break another app
- running an app can harm your entire system

OSs bear the brunt of hardware, platform complexity

- driver proliferation
- massive test labs to contend with compatibility matrix
- increasingly long release cycles

Users are de facto systems administrators

- patching, backup / recovery, application integration

# this course



# Understand browsers

Examine modern browsers and their architecture

- programming abstractions and tools, security model, extensibility model, ...

Look at recent browser developments

- industrial: NaCL, HTML5, multi-process browsers
- research: Tahoma, Gazelle, OP, SubOS, ...



# Reconsider the role of browsers

Technological determinism: browsers will continue to adopt more OS-like functions

- let's look at some of the major abstractions that OSs provide, and consider how to think of them in the Web world
  - file systems and storage
  - device support and device drivers
  - the "application" abstraction and process model
  - IPC, RPC
  - security, trust, privacy
- examine distributed OSs and programming frameworks in comparison



# Whither desktop OSs?

We'll consider alternative OS architectures assuming that Web applications dominate

- peek at some historical guesses that fell by the wayside (Java, mobile agents/objects, ...)
- look at some glimmers from industry: ChromeOS, CrunchPad, NaCL/Xax, Palm WebOS, .NET

# Administrivia

Your duties

- if you are auditing
  - sign up for mailing list, read papers before each class
- if you are registered
  - same as above
  - submit paper summaries before each class
  - sign up to lead one discussion for the quarter
  - a few small programming/design assignments



# it's your turn to talk...



# Which of these will / should win?

the XTerminal model

- apps run in your local data center, your phone/laptop/desktop exploit ample bandwidth to do remote display.

the [ Moka5 / NaCL / Xax / Tahoma ] model

- desktop OS as VMM or low-level sandbox, Web app contains rest of OS plus app. Cache code locally for latency and offline execution.

the path of least resistance

- HTML5 + RIA extensions, the browser is just an app like any other.

the [ ChromeOS / crunchpad ] model

- as above, but strip down desktop OS to bare minimum to run browser. Hide the OS as best you can, but the browser is largely unchanged.