

Welcome to CS 241 Systems Programming at Illinois

Robin Kravets

[The Team]

- Robin Kravets

- Office: 3114 SC
- rhk@illinois.edu

- TAs

- Wade Fagen, Farhana Ashraf, Hilfi Alkaff and Mainak Ghosh

- Discussion Sections

- 8 sessions (Thursdays 9, 10, 11, 12, 1, 2, 3, 4)
- All sections in SC 0220



[News and Email]

- Announcements and discussions: Piazza
 - <http://www.piazza.com/illinois/cs241>
 - All class questions
 - This is your one-stop help-line!
 - Will get answer < 24 hours
- e-mail
 - cs241help-fa12@cs.illinois.edu
 - Personal questions not postable on the news group



[The Textbook]

- Introduction to Systems Concepts and Systems Programming
 - University of Illinois Custom Edition
 - Copyright © 2007
 - Pearson Custom Publishing
 - ISBN 0-536-48928-9

- Taken from:
 - Operating Systems: Internals and Design Principles, Fifth Edition, by William Stallings
 - UNIX™ Systems Programming: Communication, Concurrency, and Threads, by Kay A. Robbins and Steven Robbins
 - Computer Systems: A Programmer's Perspective, by Randal E. Bryant and David R. O'Hallaron



[Your CS 241 “Mission”]

- Come to class
 - MWF, 10-10:50am
 - Please participate actively...
 - Attend 1 discussion section per week
- Read textbook
 - Reading assignments posted on webpage
- Homework (1) 3%
- Programming assignments (8) 47%
 - Longer MPs are worth a little more
- Midterm 20%
 - 7:00 – 9:00 PM October 15th
- Final 30%
 - 8:00-11:00 AM, December 17



[It's all about the programming!]

■ MPs

- Goal
 - Expose you to the concepts and APIs taught in class
- All individual
 - You can't learn it if you don't do it yourself!

■ MP Contest

- Memory (`malloc`)
- Prizes and bragging rights

■ Components for grading

- Correctness
 - Autograder
 - Once a night to help you check correctness
 - Does not reflect grade
- Memory
 - `valgrind`
- Debugging
 - `gdb`
- Knowing your code
 - 1 page write-up (6 MPs)
 - Oral description (1 MP)



[Deadlines]

- Homework

- Deadlines are strict
- Late submissions will not be considered

- MPs

- Please respect posted deadlines to ensure quick grading
- Late MPs will be penalized 2% for each late hour (rounded off to the higher hour)
- No submissions past 48 hours



[Regrades]

- Considered if you were graded incorrectly
- Within one week of posting of grades for a quiz, homework, MP or exam
- Regrades must be submitted in writing on a separate piece of paper
 - Please do not write on your homework, MP or Exam



[Academic Honesty]

- Your work in this class **must** be your own.
- If students are found to have cheated (e.g., by copying or sharing answers during an examination or sharing code for the project), **all** involved will at a minimum receive grades of 0 for the first infraction and reported to the academic office.
- Further infractions will result in failure in the course and/or recommendation for dismissal from the university.
- Department honor code:
<https://wiki.engr.illinois.edu/display/undergradProg/Honor+Code>



What is cheating in a programming class?

- At a minimum
 - Copying code
 - Copying pseudo-code
 - Copying flow charts
- Consider
 - Did some one else tell you how to do it?
- Does this mean I can't help my friend?
 - No, but don't solve their problems for them
- Not cheating
 - Discussing high-level approaches
 - Discussing MP requirements, C language, tools
 - Helping each other with debugging



Getting The Most Out Of Any Class

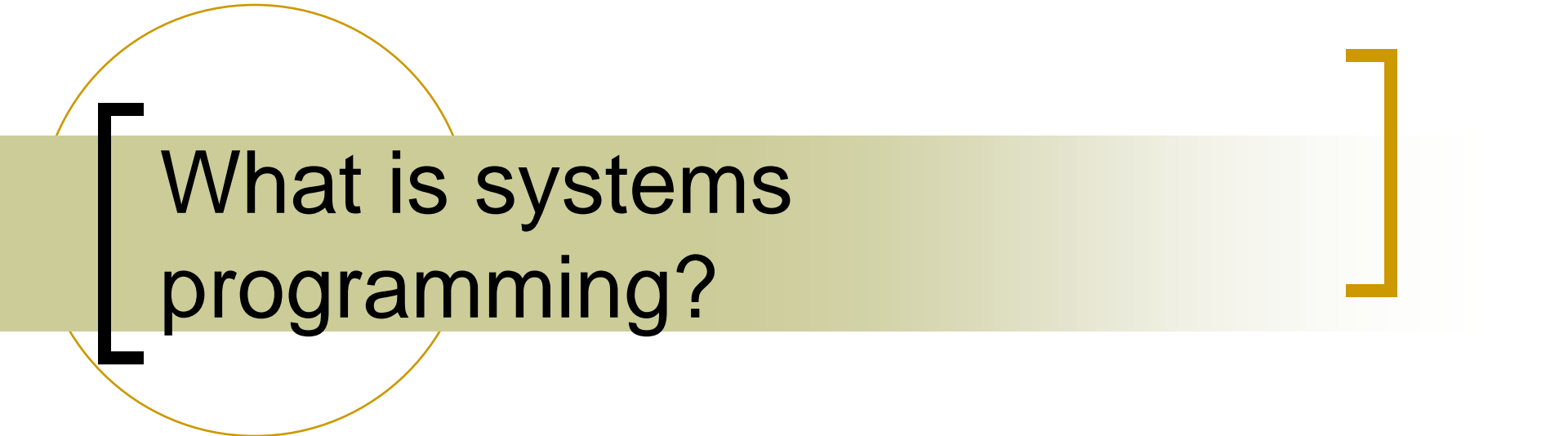
- Get the big picture
 - Why are we doing this?
 - Why is it important?
- Understand the basic principles
 - If you know how to apply them, you can work out the details
- Learn why things work a certain way
 - Automatic vs. manual, elegant vs. ad hoc, solved problem vs. open
- Think about the cost-benefit trade-offs
 - Performance vs. correctness, development time vs. benefit



Getting The Most Out Of This Class

- Attend the lectures, use videos as a back up
- Pay attention to the discussions
- Ask questions, and participate
- Do the exercises in class
- Start the assignment the day it's handed out, not the day it's due





What is systems programming?

What is systems programming?

***sys·tem* Noun /'sistəm/**

1. A set of connected things or parts forming a larger and more complex whole.

2. An integrated set of elements that accomplish a defined objective

- Examples: Digestive system, economic system, ecosystem, social systems
- Computer systems: collections of programs
 - Search engines, social networks, databases, Internet
 - In this class, we learn how to design and implement computer systems



Challenges in programming computer systems

- Sharing resources among programs
- Preventing malicious/incorrect programs from interfering with other programs
- Coordinating operations of multiple programs
- Communicating information between programs

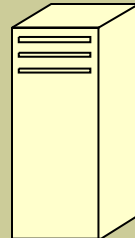
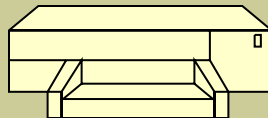
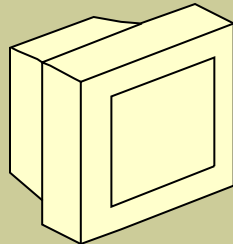


What is an operating system and why do I need one?

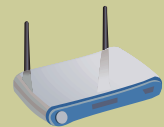
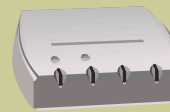
- What do we have?
 - Set of common resources

My Computer

Hardware



Network

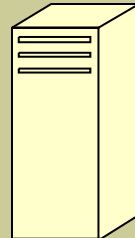
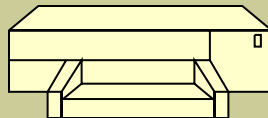
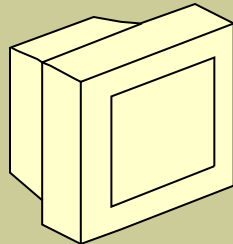


What is an operating system and why do I need one?

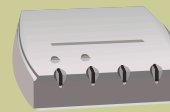
- What do we have?
 - Set of common resources
- What do we need?

My Computer

Hardware



Network



What is an operating system and why do I need one?

Application Software

Firefox

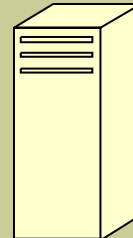
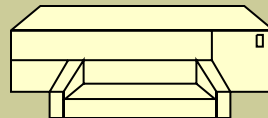
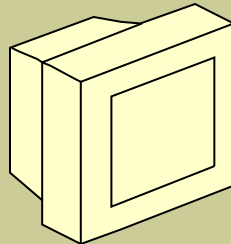
Second Life

Yahoo
Chat

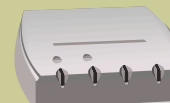
GMail

- A clean way to allow applications to use these resources!

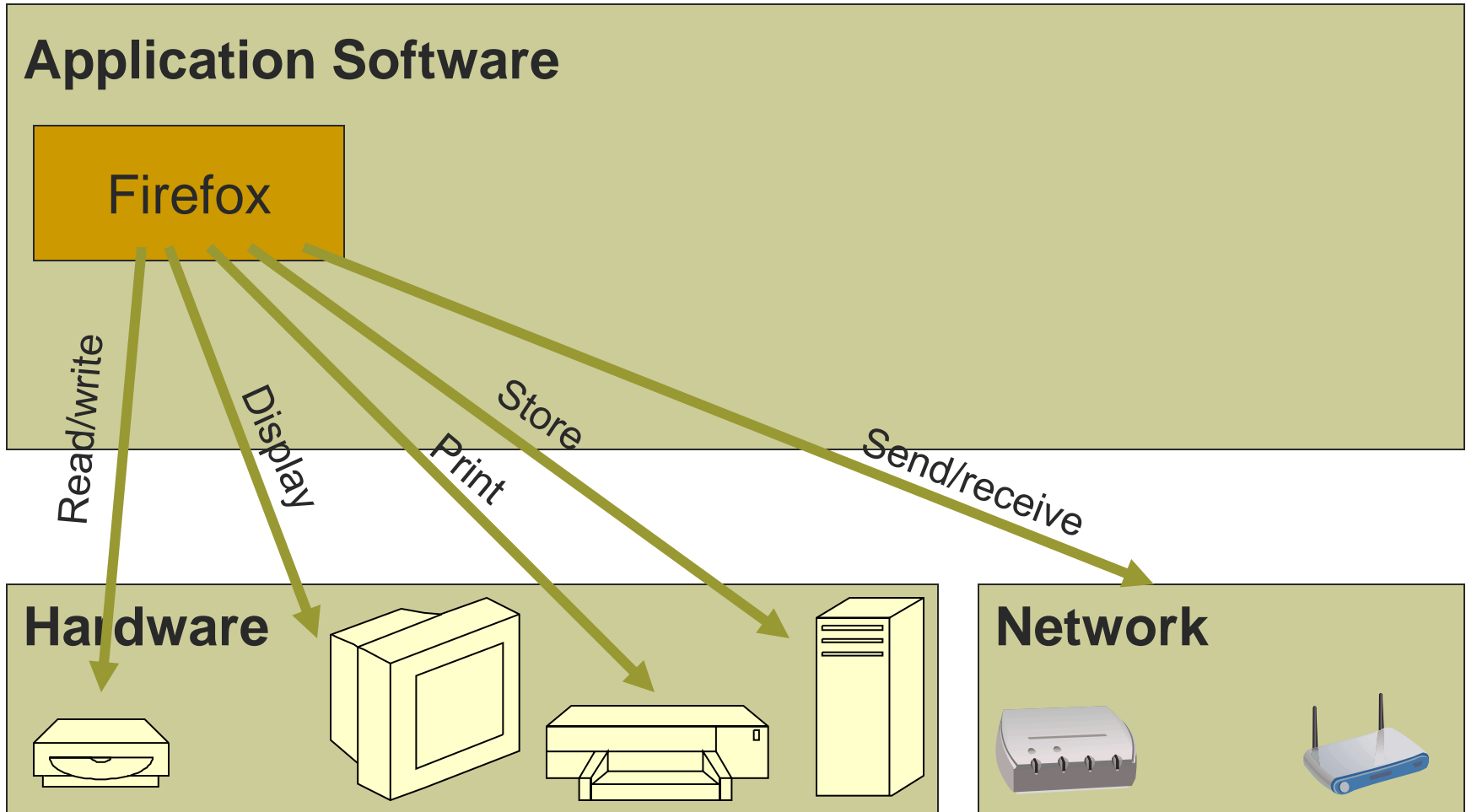
Hardware



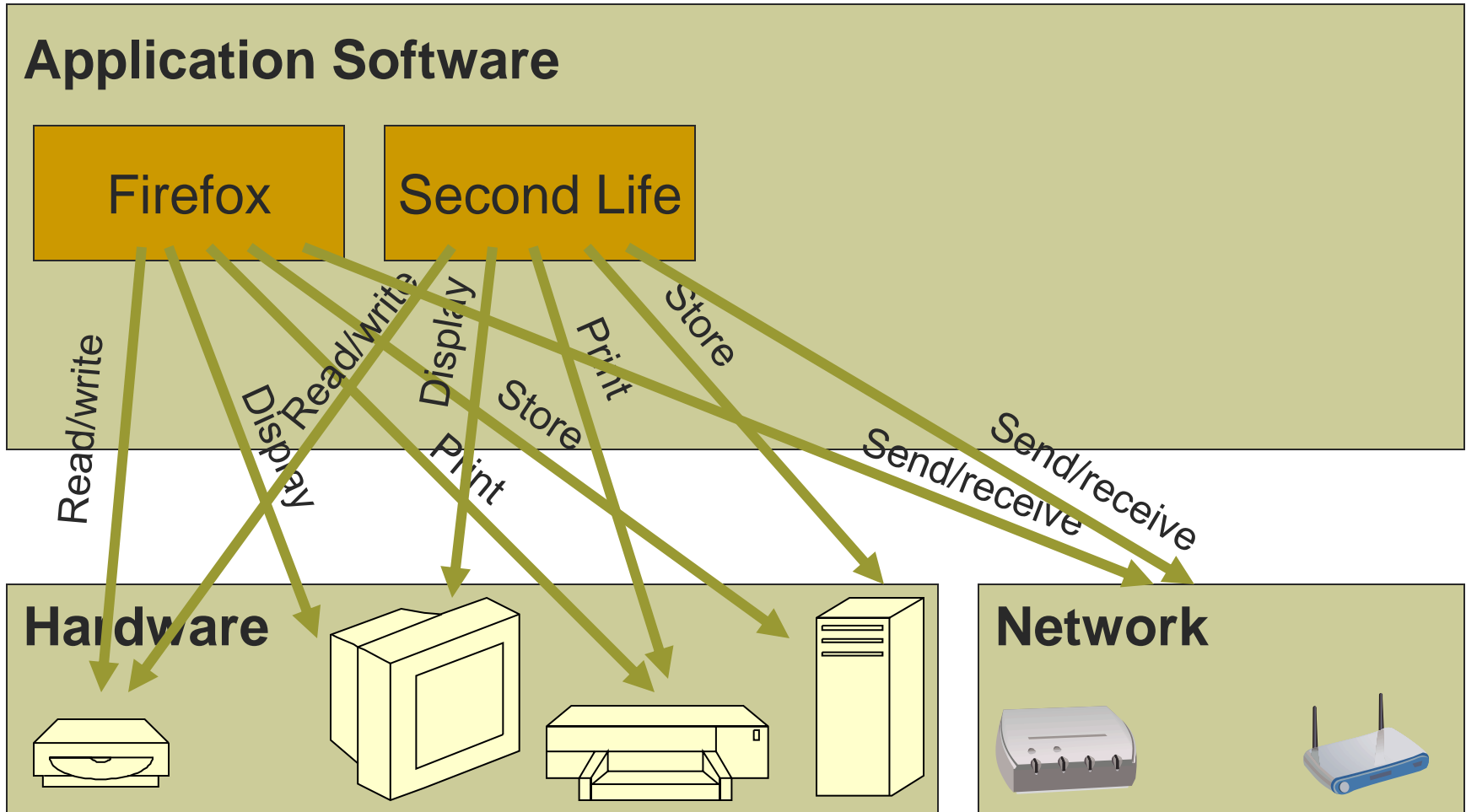
Network



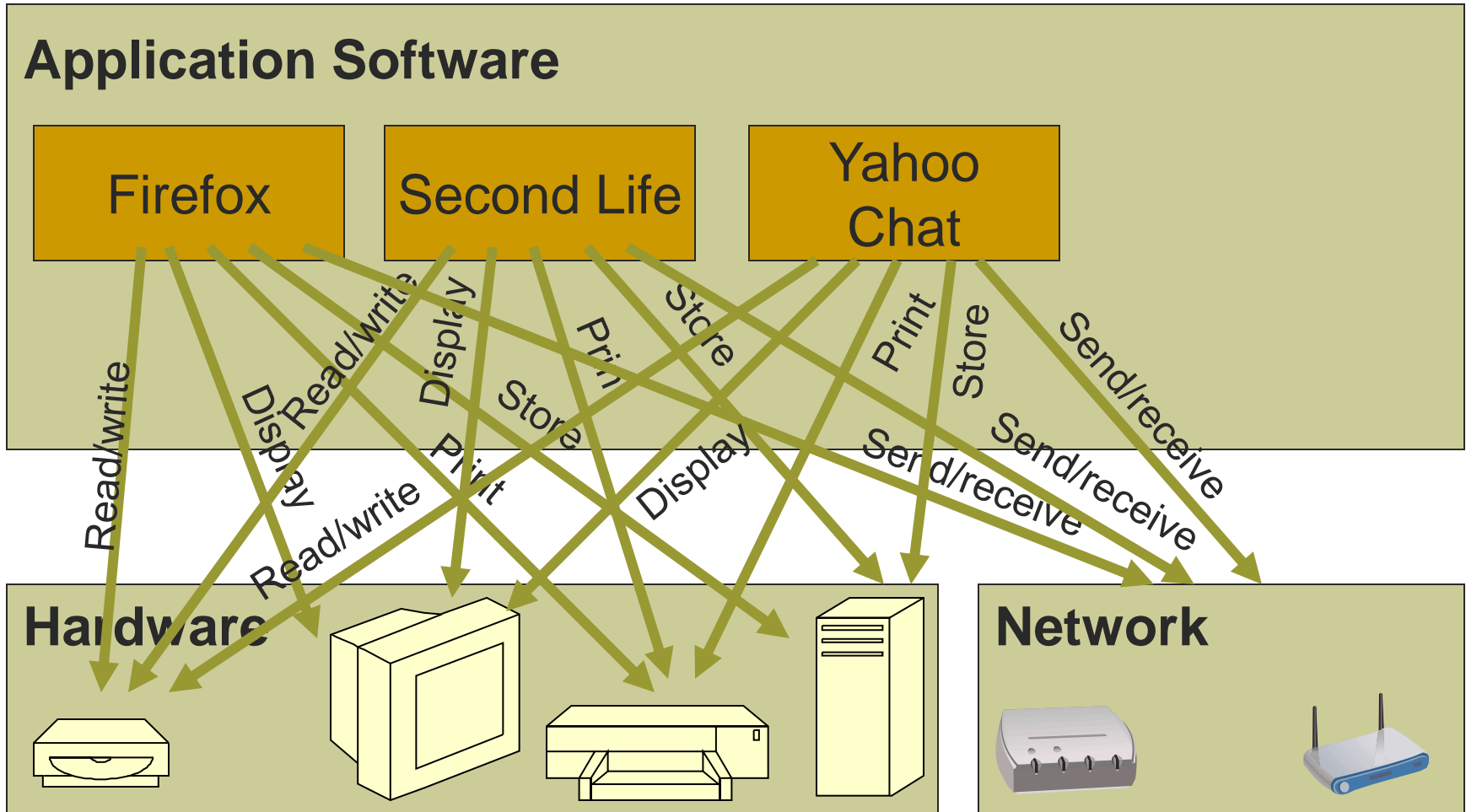
Application Requirements



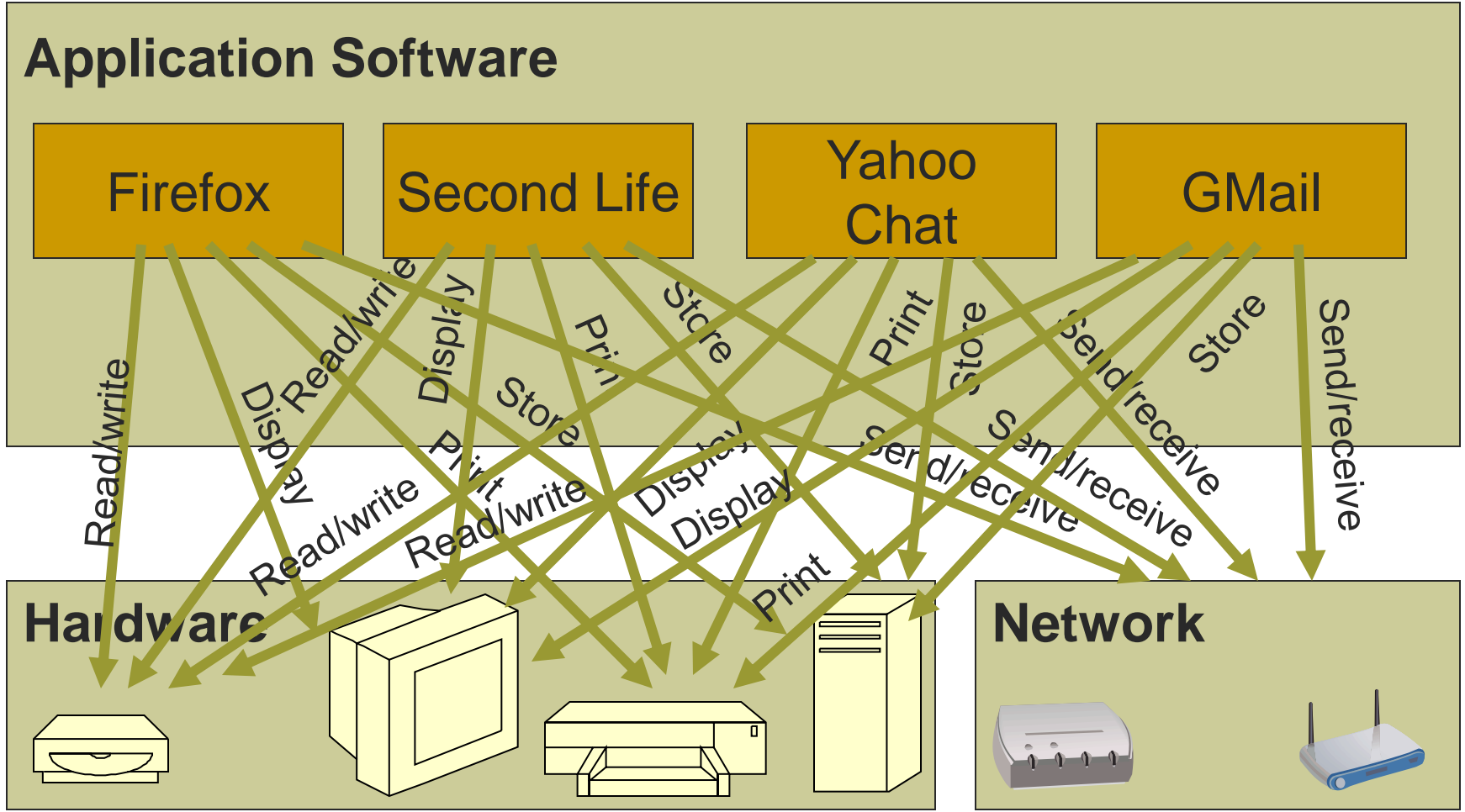
[Two Applications?]



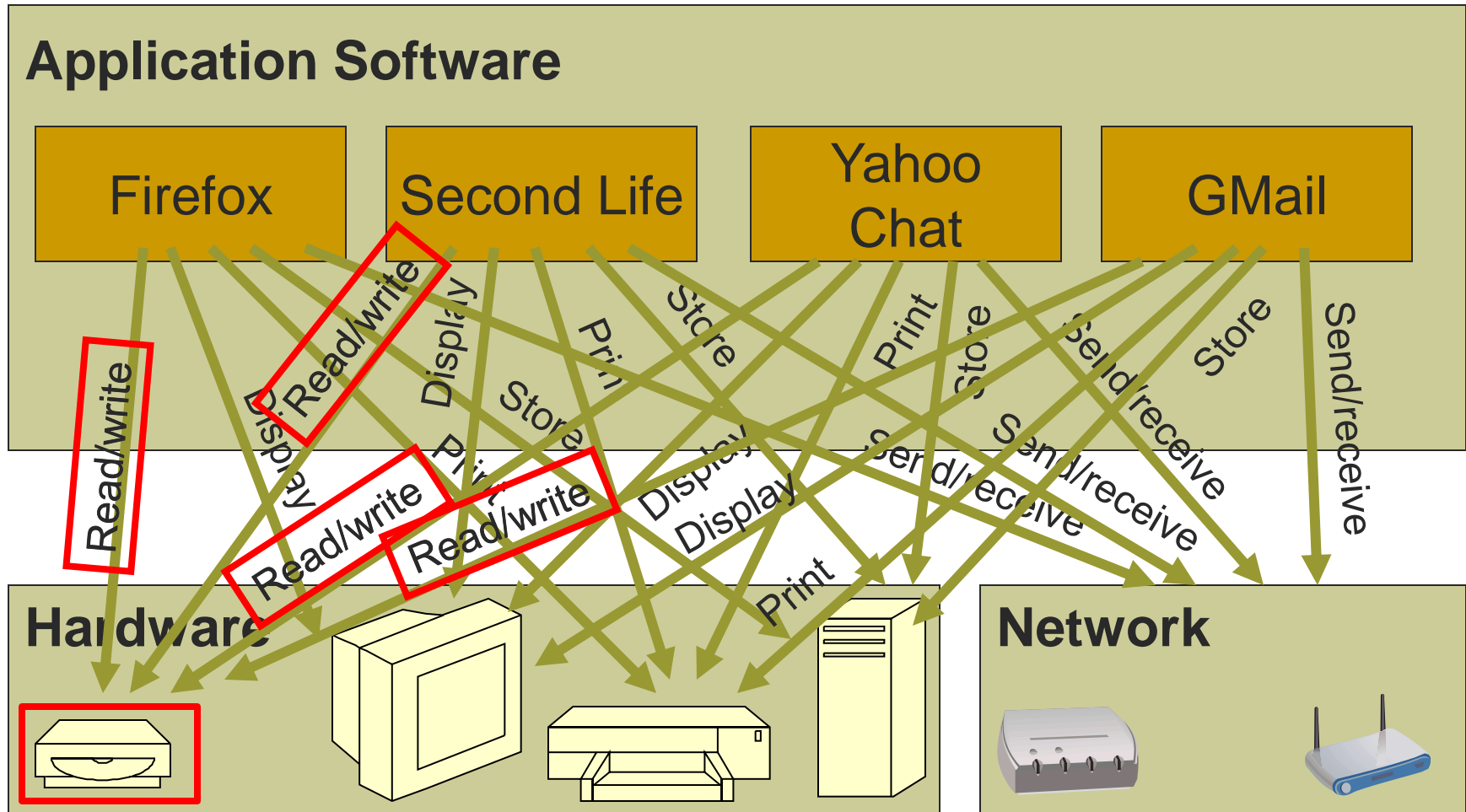
Managing More Applications?



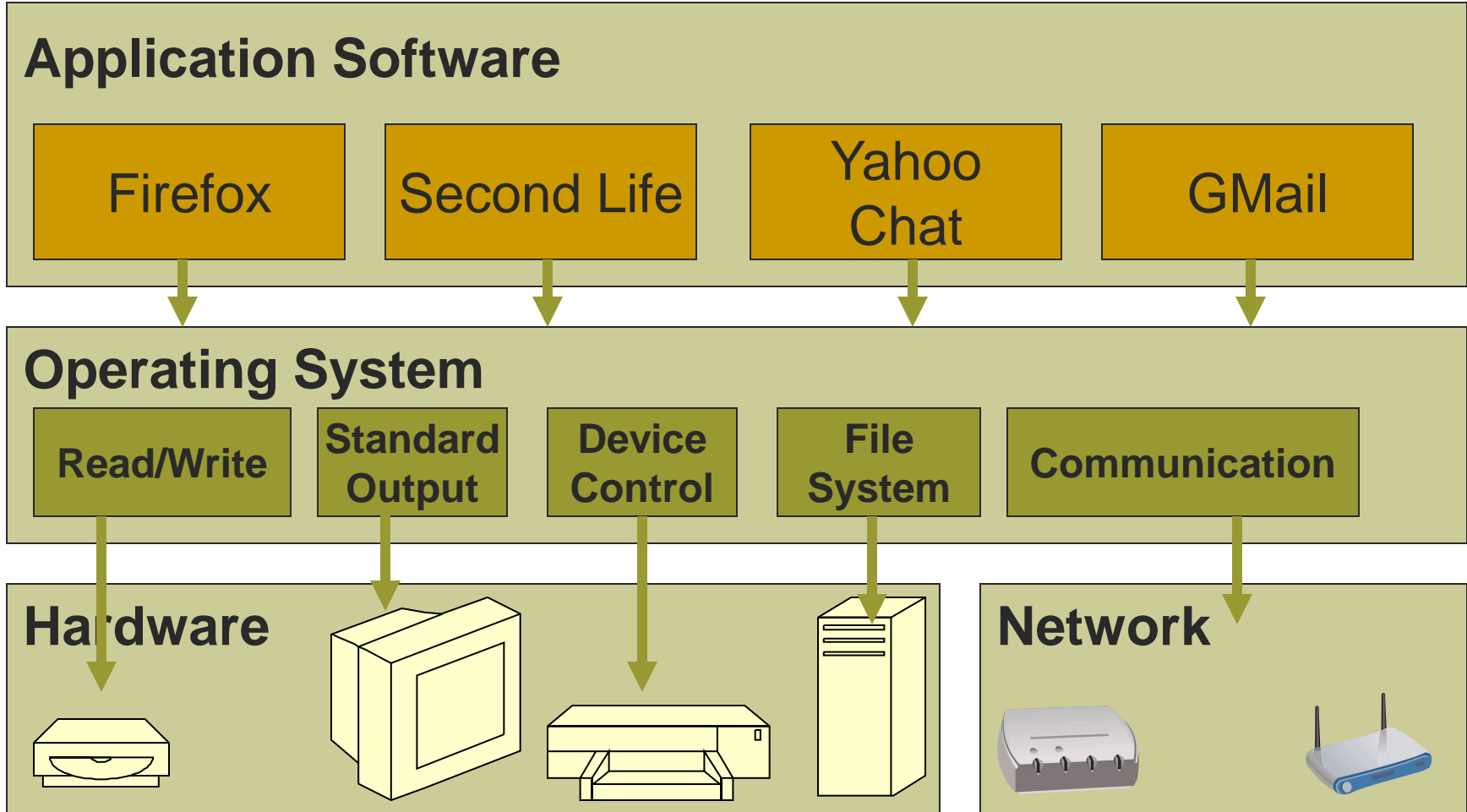
[We need help!]



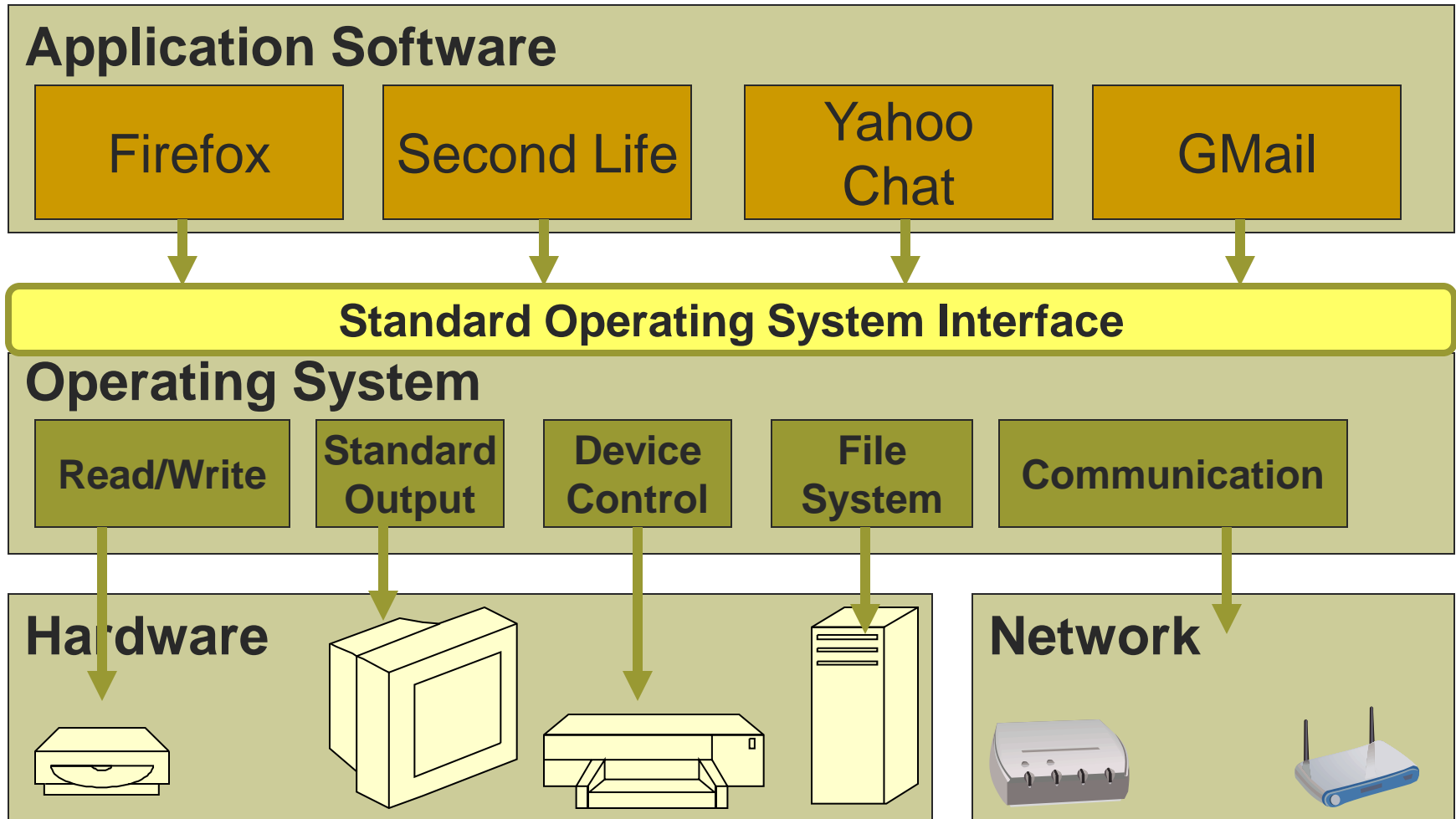
Approach: Find Common Functions



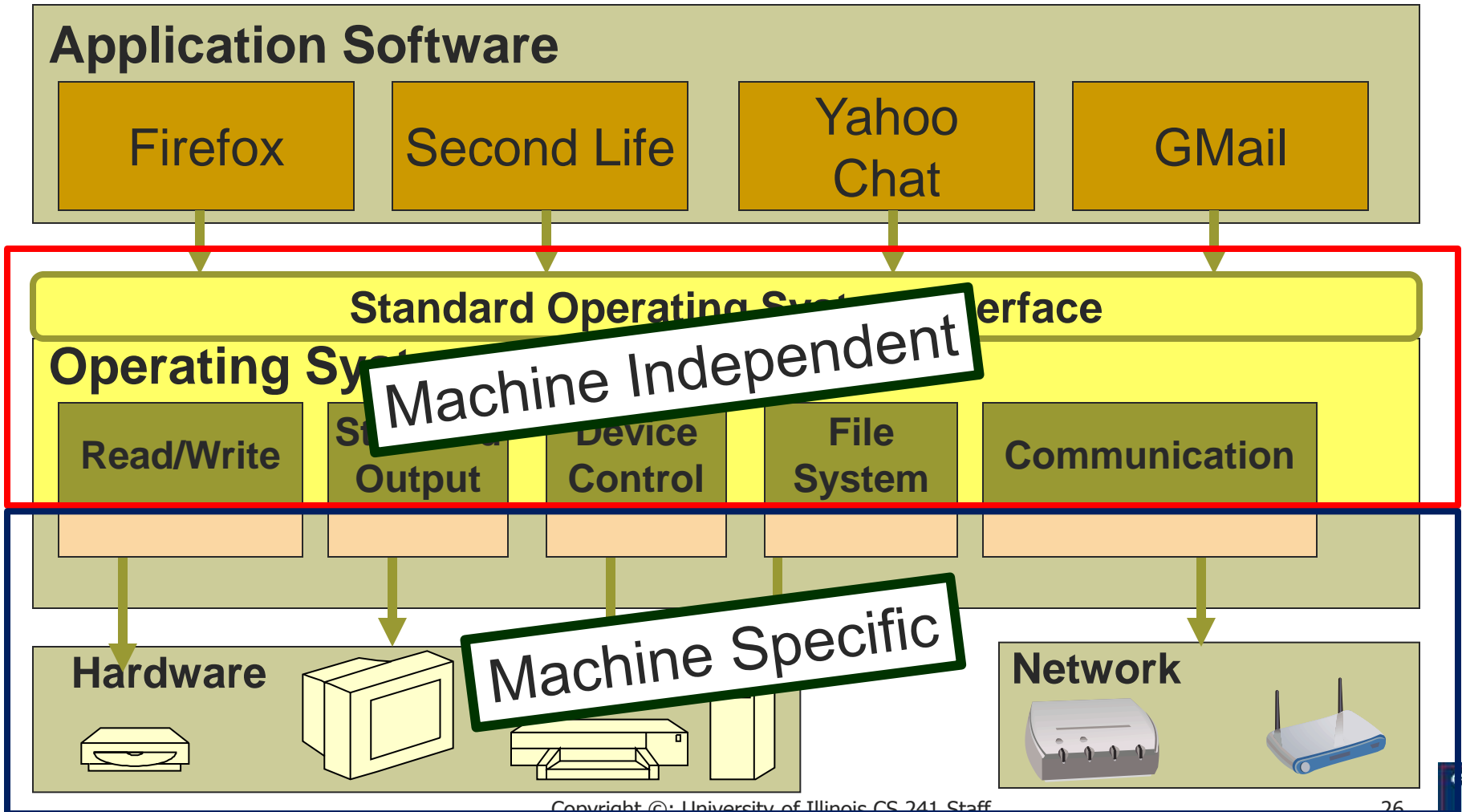
[Delegate Common Functions]



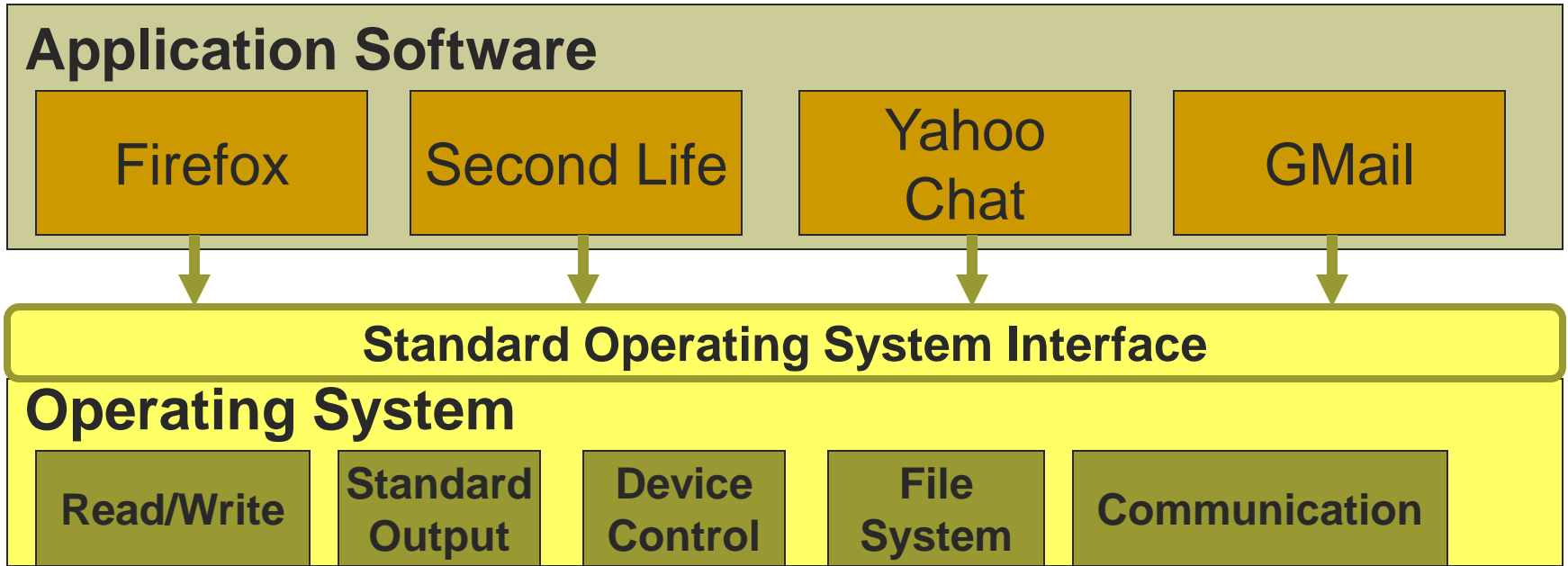
[Export a Standard Interface]



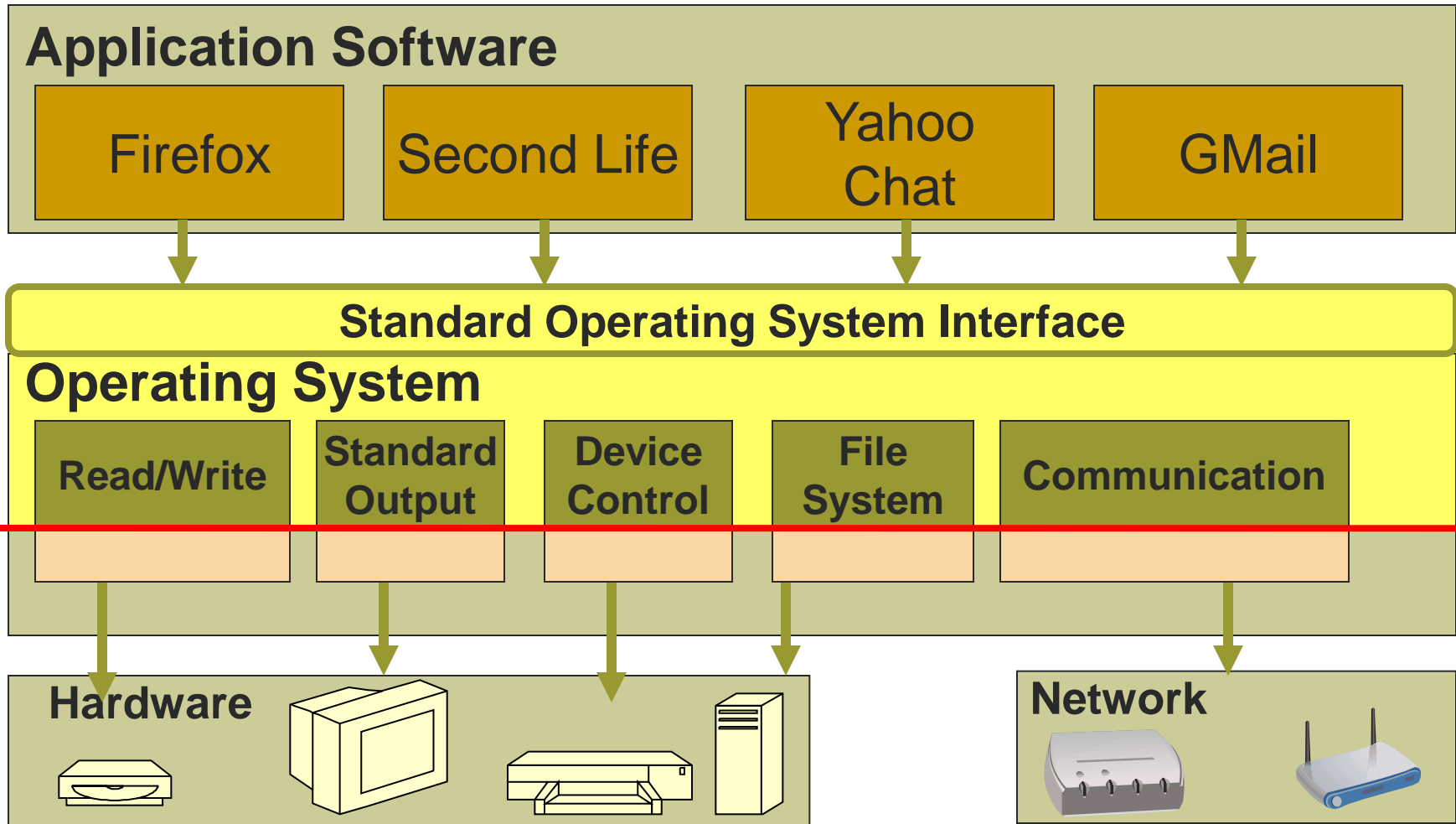
Goal: Increase Portability



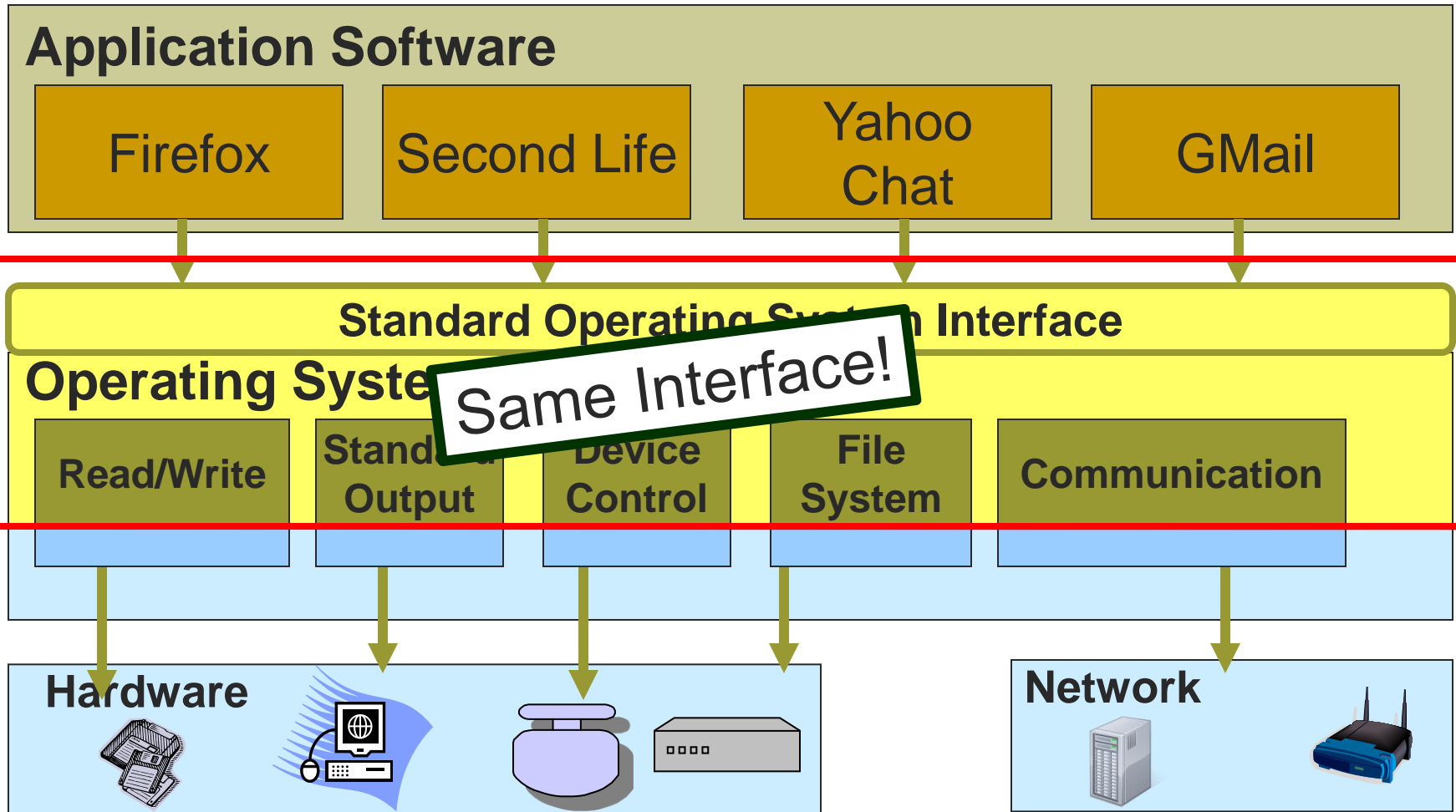
[Machine Independent = Portable]



OS Runs on Multiple Platforms

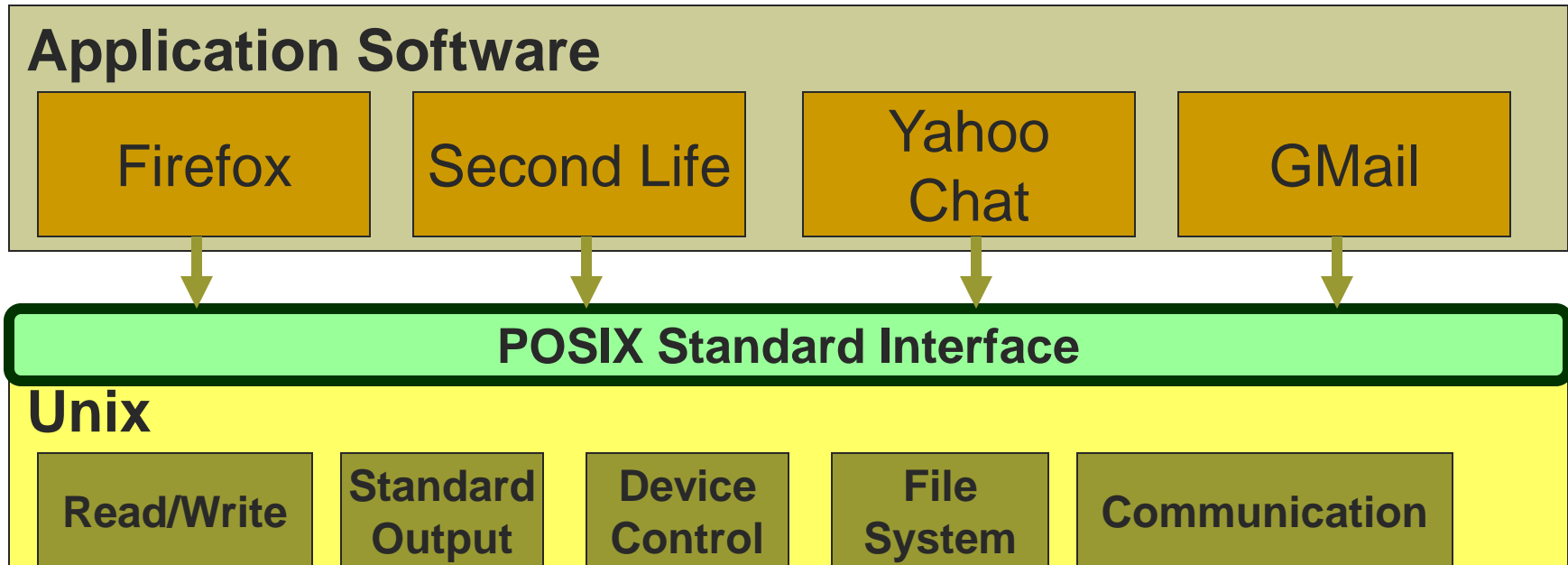


OS Runs on Multiple Platforms



POSIX

The UNIX Interface Standard



[Big goal: modularity]

- **Modularity:** Decomposition of a large task into smaller reusable components with well-known interfaces between them
- Advantages
 - Simplicity
 - Portability
 - Re-use common functions
 - Abstraction: hide details of implementation



Course Questions

- What are the right abstractions and interfaces to let pieces of a system work together smoothly?
- ...and how do I use them?
- What goes on “behind the scenes” in interfaces I’ve been using?
 - Memory, files, network, ...
- How do we tame the complexity of a big system?
 - “Systems programming” is a lot more than just programming!



[Course Objectives]

- By the end of this course, you should be able to:
 - Identify the basic components of an operating system
 - Describe their purpose
 - Explain the “black box” abstract interface and how they function “inside the box”
- Use the system effectively
 - Write, compile, debug, and execute C programs
 - Correctly use system interfaces provided by UNIX (or a UNIX-like operating system)
- Build your own large, multi-process, networked applications



[Course Outline]

- Week 1-2: **Nuts & bolts**
 - Manipulate pointers and memory
 - Use UNIX system calls from within C programs
 - MP1: working with C pointers & strings
- Week 3-4: **Memory**
 - Understand memory allocation and virtualization
 - MP2: malloc (+contest!)



Course outline

- Week 5-6: **Parallelism**
 - Create and manage processes and threads
 - Control scheduling of proc./threads
 - MP3: Shell
 - MP4: Multithreaded sorting
 - MP5: Scheduling algorithm simulator
- Week 7-11: **Cooperating parallelism**
 - Communicating & sharing resources between proc./threads
 - MP6: Parallel make
 - MP7: MapReduce



Course outline

- Week 12-13: **Networking**
 - Use communication protocols (TCP/IP) and interfaces (Sockets)
 - Write distributed multi-threaded apps that talk across a network
 - MP8: Web server (*)
- Week 14: **Additional OS concepts**
 - I/O and file systems



[Complete Schedule]

- See class webpage
 - <http://courses.engr.illinois.edu/cs241/>
 - Schedule is dynamic
 - Check regularly for updates
- Slides will be posted by the night before class
 - Bring a print out of the slides to class
 - Some class material may not be in slides
 - Examples may be worked out in class



[Your to-do List]

- Visit the class webpage
 - Check out all the info
 - Especially schedule, grading policy, homework & MP hand-in instructions, and resources
- Familiarize yourself with Piazza
- Find a reference to refresh your C programming skills
 - <http://www.lysator.liu.se/c/bwk-tutor.html>

