

# Welcome to CS 241 Systems Programming at Illinois

Instructor: Brighten Godfrey

# [The Team]

- Brighten Godfrey
  - Office: 3211 SC
  - Office hour this week: Fri 1:30 – 2:30
  - [pbq@illinois.edu](mailto:pbq@illinois.edu)
- TAs
  - Wade Fagen, Farhana Ashraf, Matt Trower, Yunlong Gao
- Discussion Sections
  - 8 sessions (Thursdays 9, 10, 12, 1, 2, 3, 4, 5)
  - All sections in SC 0220
  - **Please move out of the 11:00 a.m. session**



# [ News and Email ]

- Piazza for announcements and discussions
  - <http://www.piazza.com/illinois/cs241>
  - This is your one-stop help-line!
  - Will get answer < 24 hours
- Email
  - [cs241help-sp12@cs.illinois.edu](mailto:cs241help-sp12@cs.illinois.edu)
  - Only for personal questions not postable on Piazza



# [The Textbook]

- Introduction to Systems Concepts and Systems Programming
  - University of Illinois Custom Edition
  - ISBN 0-536-48928-9
- Taken from:
  - Operating Systems: Internals and Design Principles, Fifth Edition (Stallings)
  - UNIX™ Systems Programming: Communication, Concurrency, and Threads (Robbins & Robbins)
  - Computer Systems: A Programmer's Perspective (Bryant & O'Hallaron)



# [ Course components ]

- Come to class
  - MWF, 11-11:50am
  - Please participate actively
- Attend 1 discussion section per week
- Read textbook (assignments posted on webpage)
- Homework (1) 3%
- Programming assignments (8) 47%
- Midterm: March 6, 7-9pm 20%
- Final: time & date TBA 30%



# [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 24 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 exam or homework



# [ 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 will be 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>

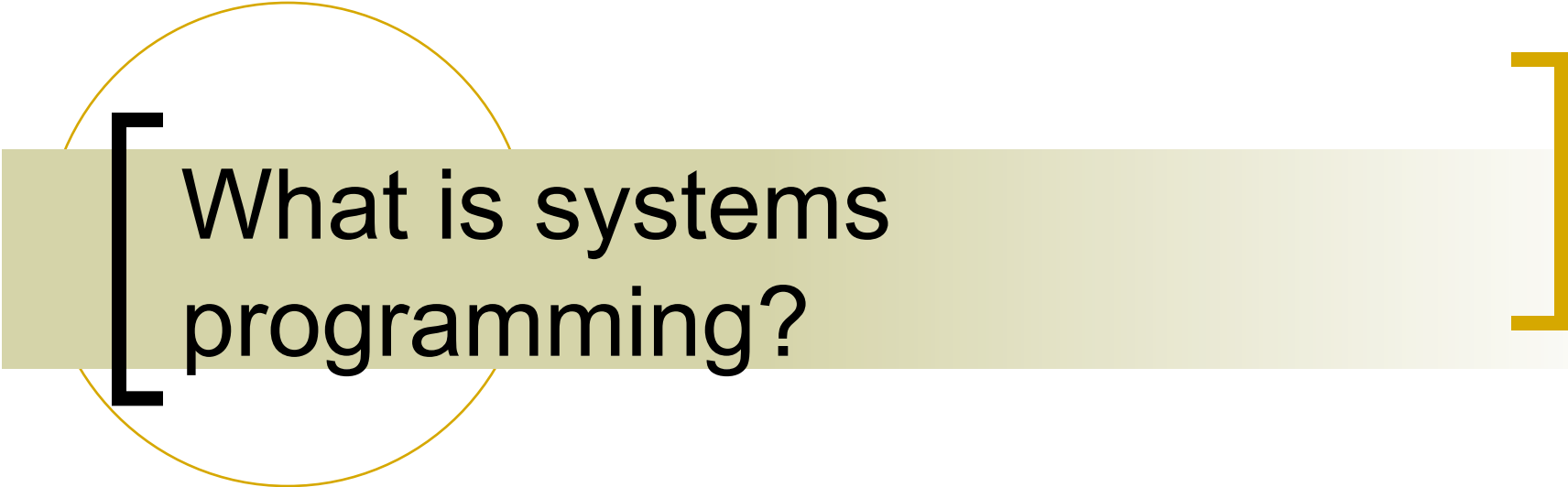




# [ Cheating vs. collaborating ]

- Cheating
  - Copying code, pseudo-code, flow charts
  - Writing someone else's code line by line
- Not cheating
  - Discussing high-level approaches
  - Discussing MP requirements, C language, tools
  - Helping each other with debugging
- Consider
  - Did some one else tell you how to do it?





What is systems programming?

# [ What is a system? ]

## ***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: Computer systems, economic system, ecosystem, social systems, digestive system, ...

■ 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 building computer systems

- **Sharing** resources among programs
- **Preventing interference** from malicious/  
incorrect programs
- **Coordinating** operations of multiple  
programs
- **Communicating** information between  
programs



# [ Interactions ]

---

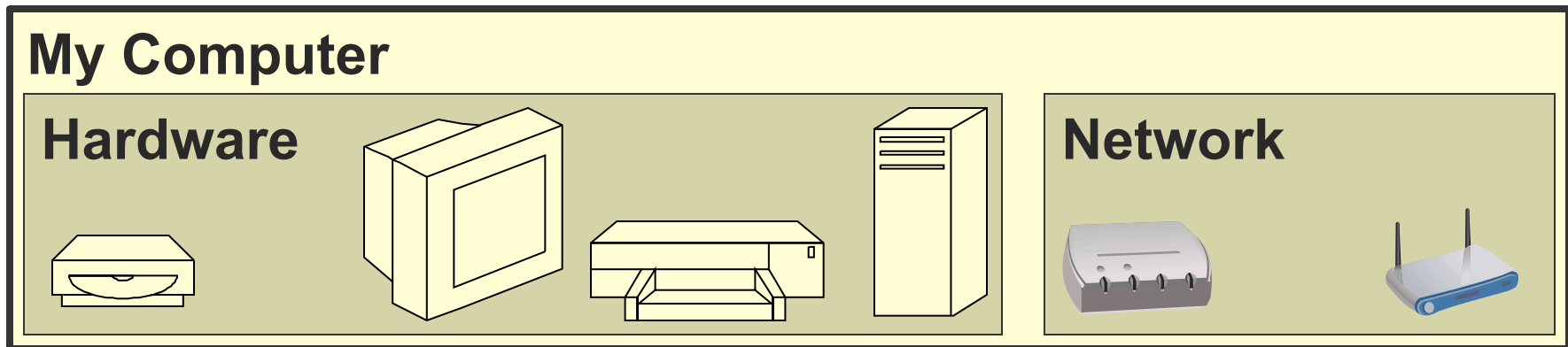
“What we are concerned with here is the fundamental interconnectedness of all things.”

– Dirk Gently



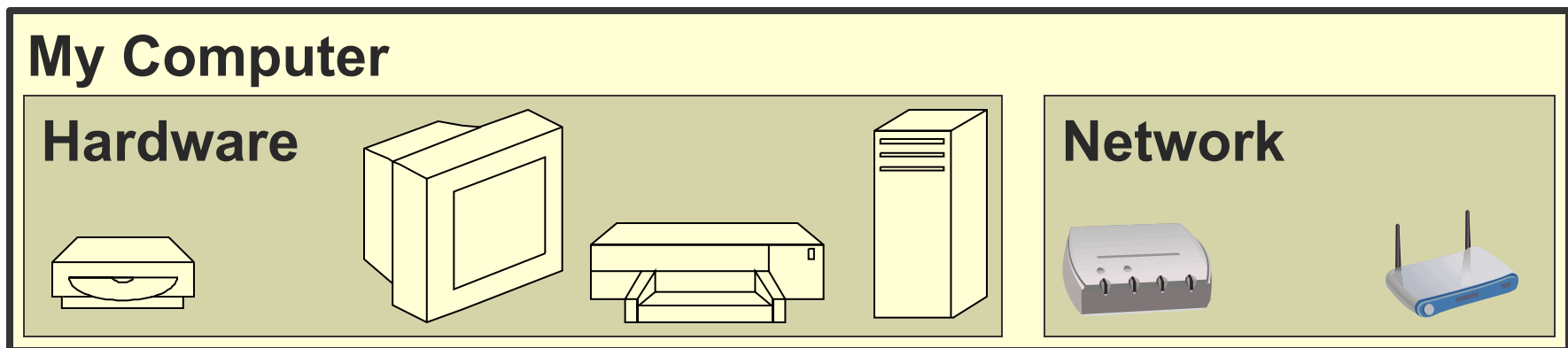
# What is an operating system and why do I need one?

- What do we have?
  - Set of common resources



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

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



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

## Application Software

Firefox

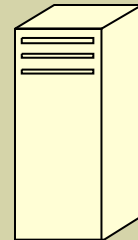
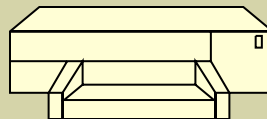
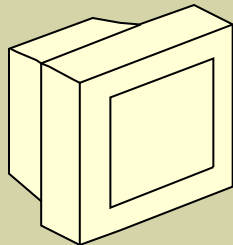
Second Life

Yahoo  
Chat

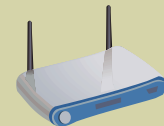
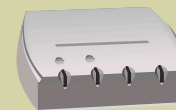
GMail

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

## Hardware

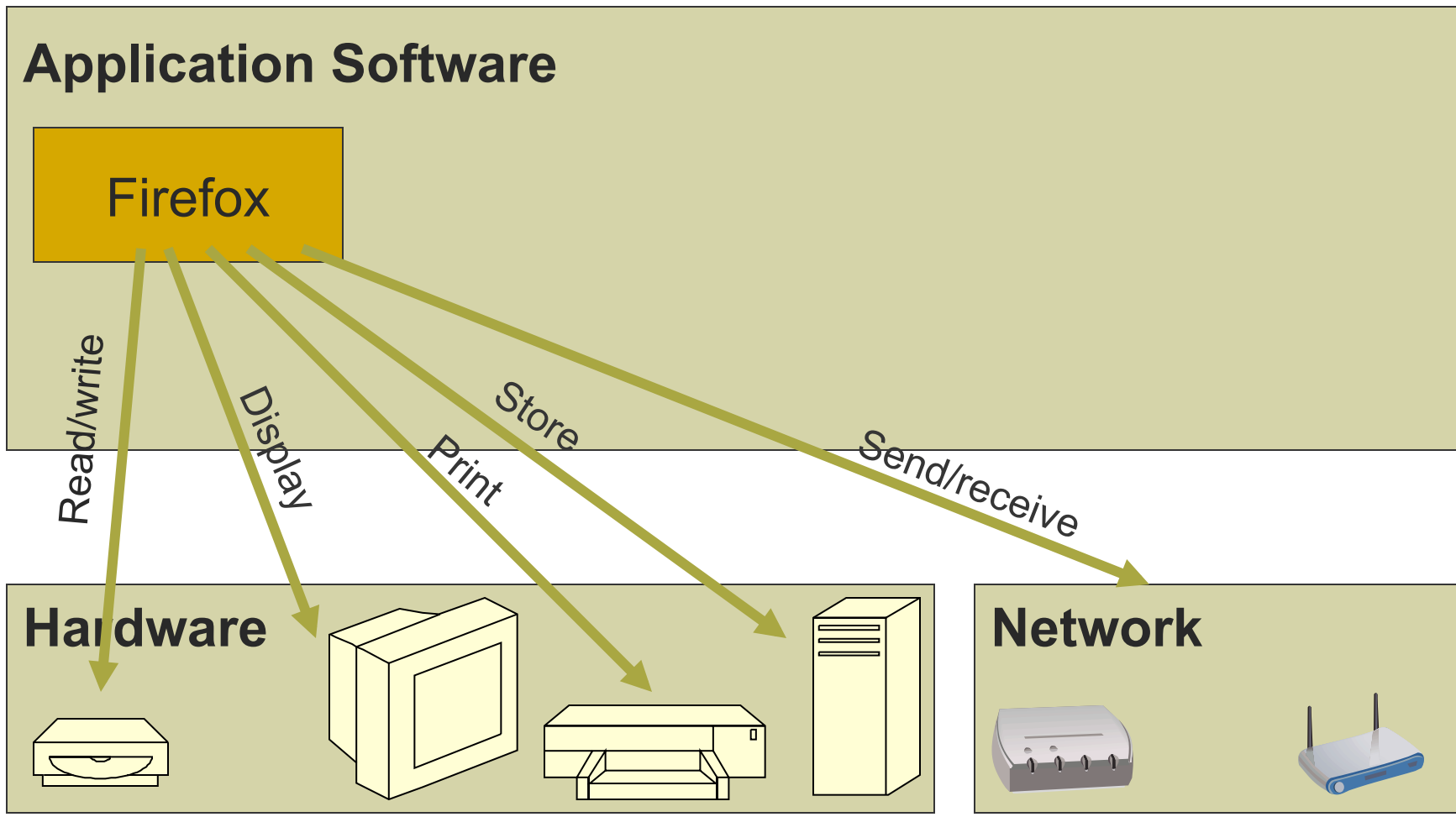


## Network

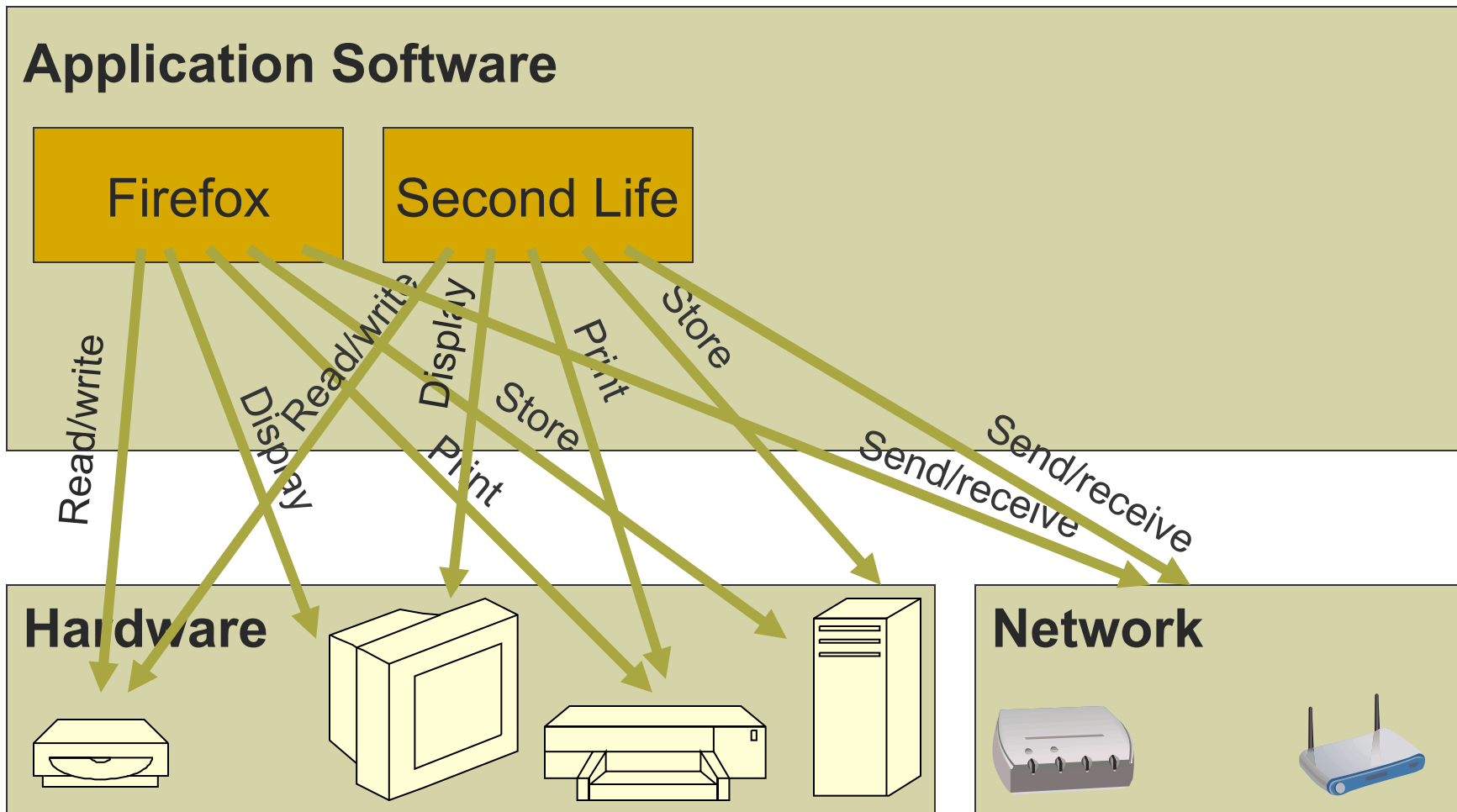




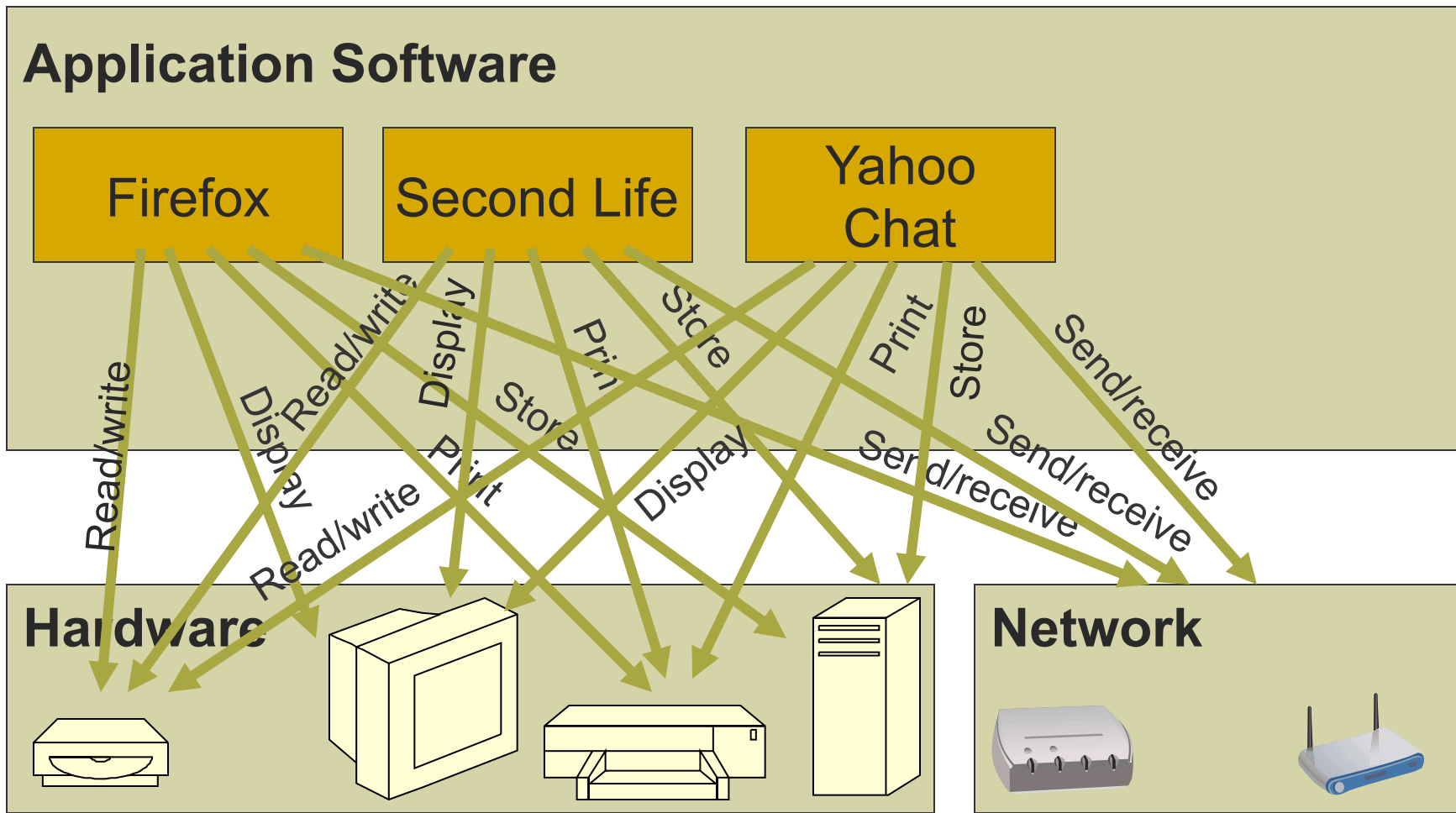
# Application Requirements



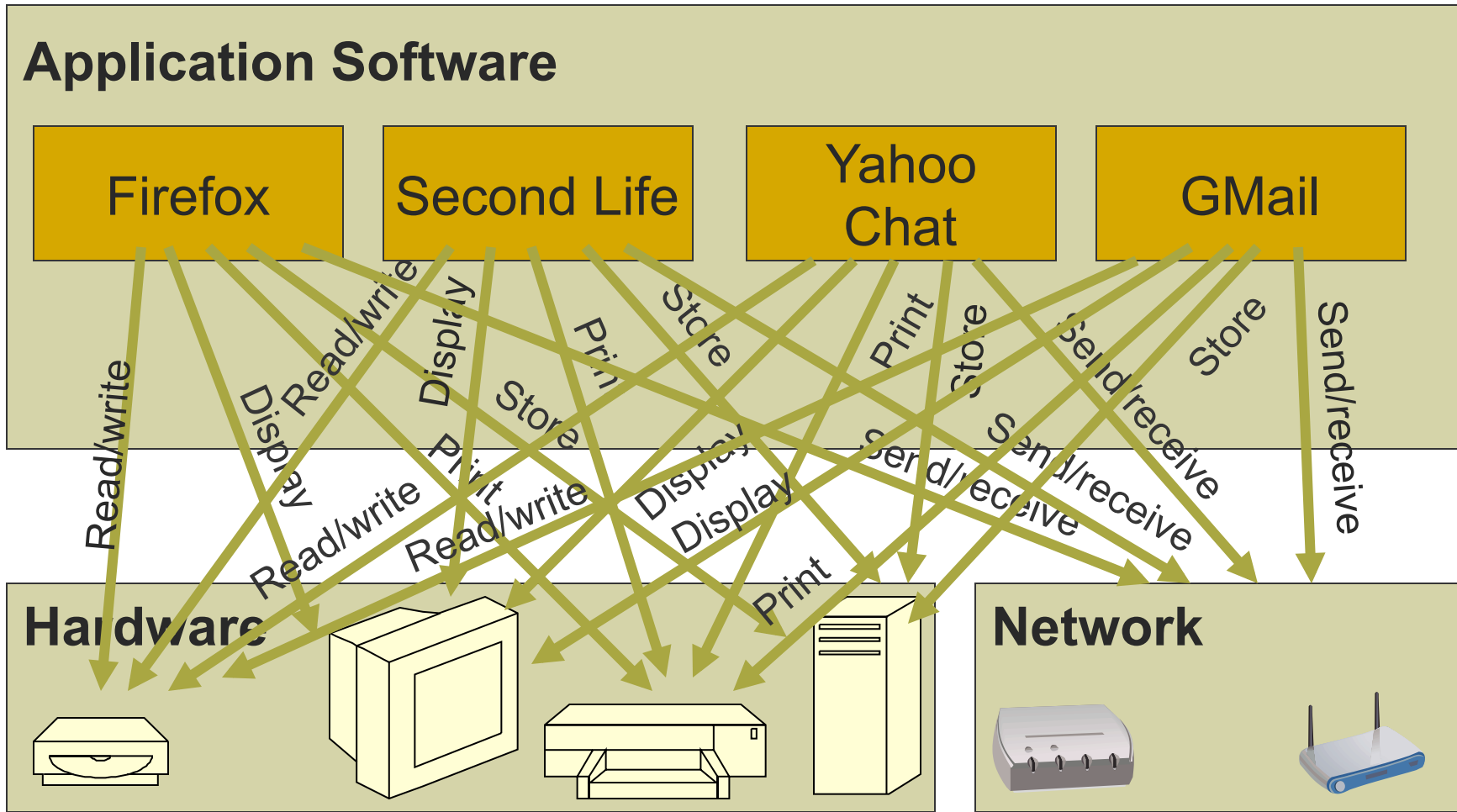
# [ Two Applications? ]



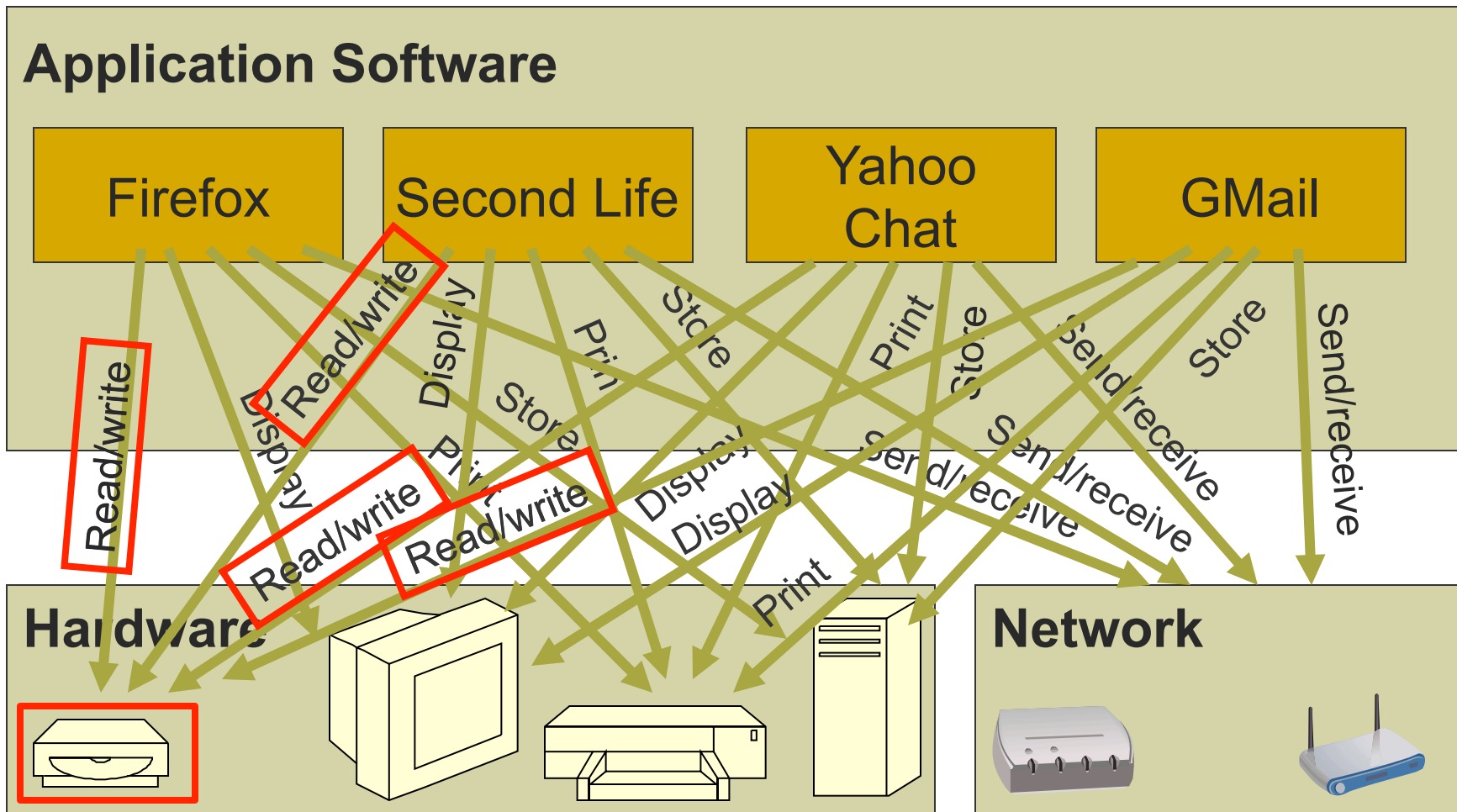
# Managing More Applications?



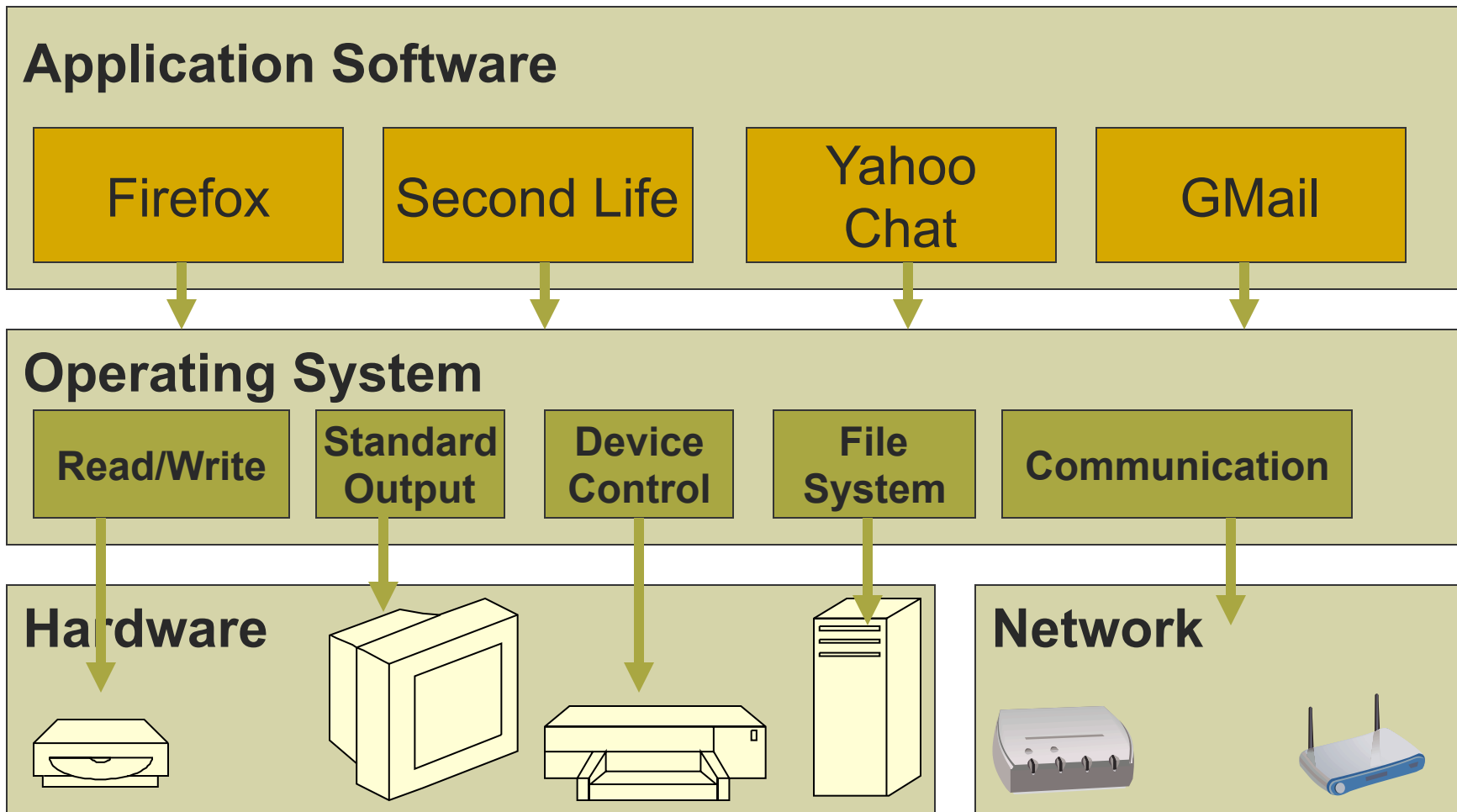
*We need help!*



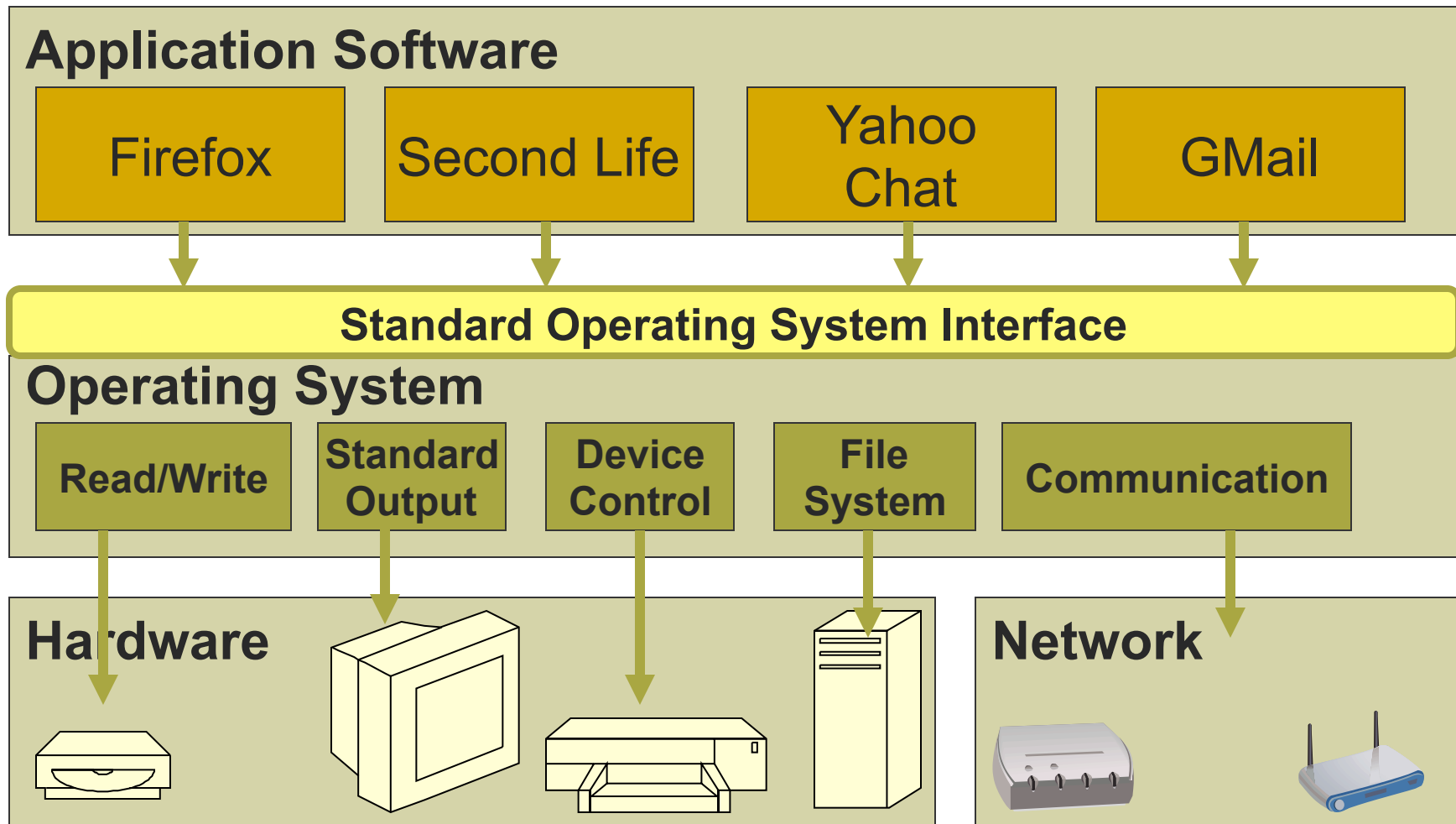
# Approach: Find Common Functions



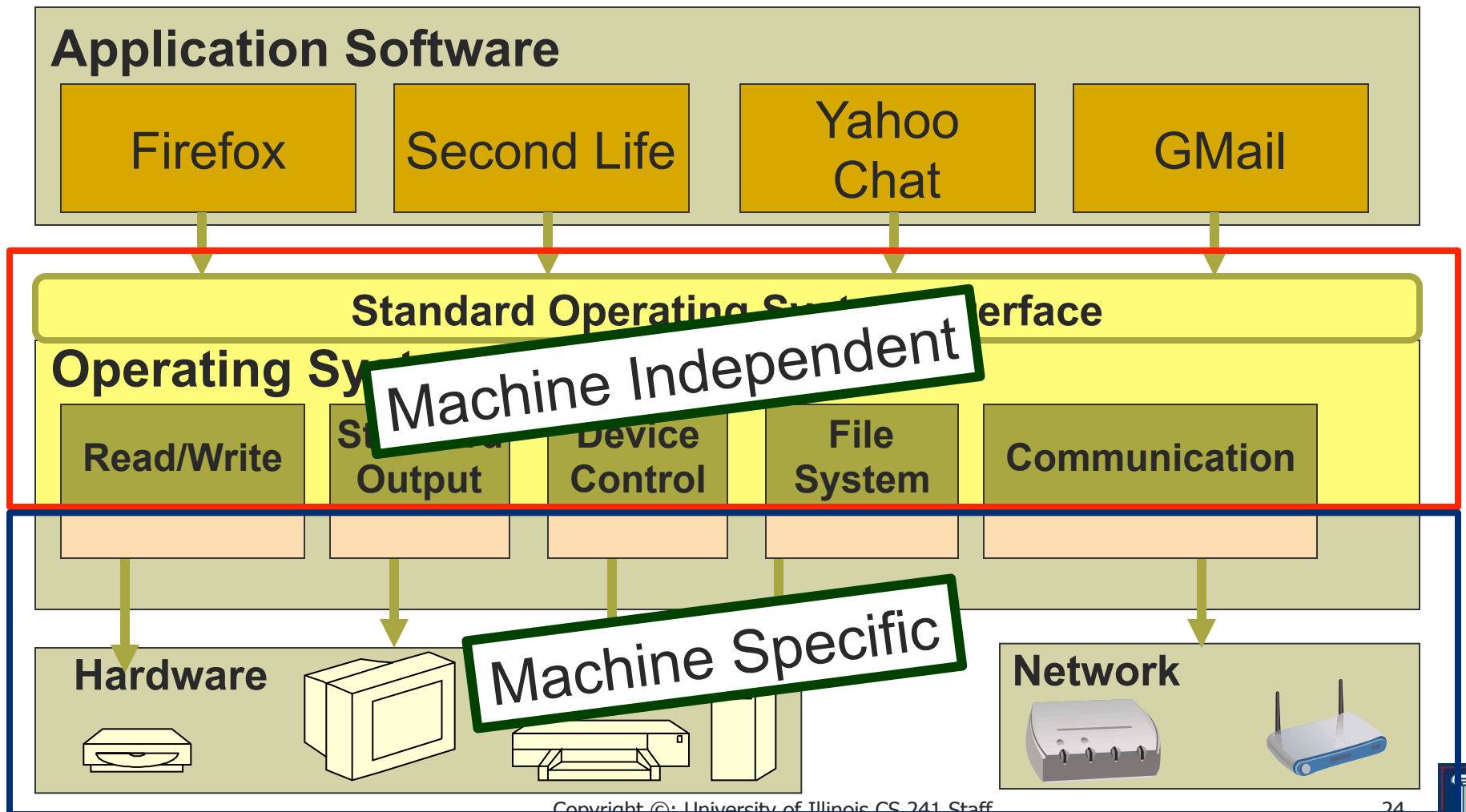
# Delegate Common Functions



# Export a Standard Interface

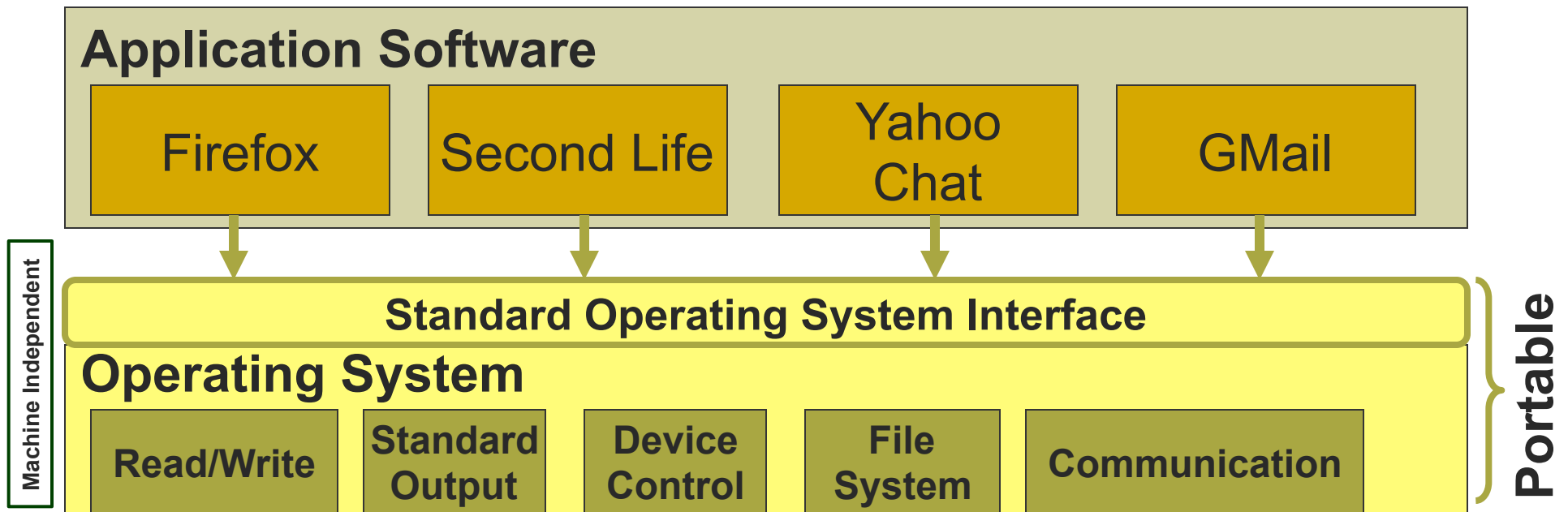


[ One goal: Increase portability ]

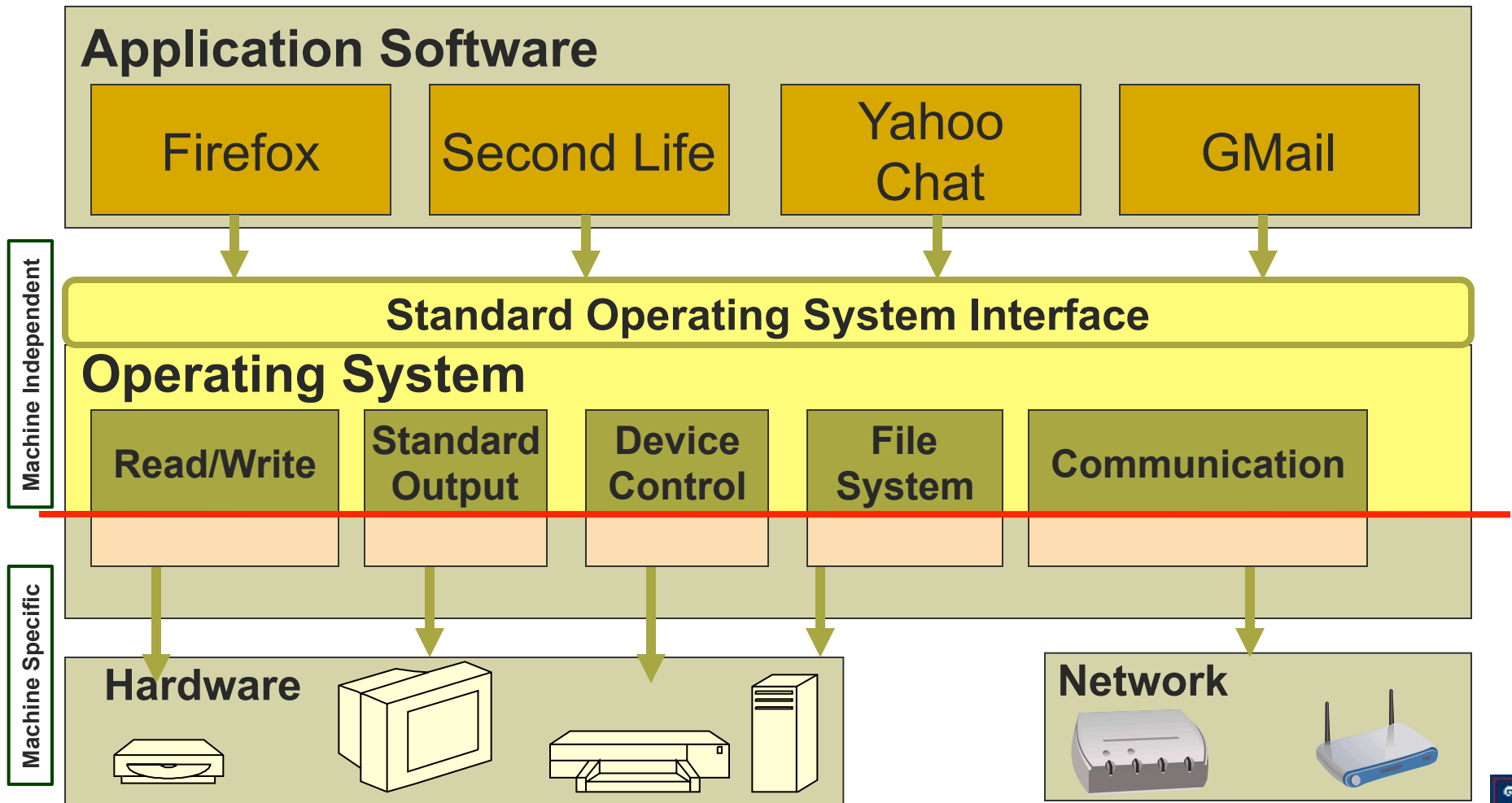




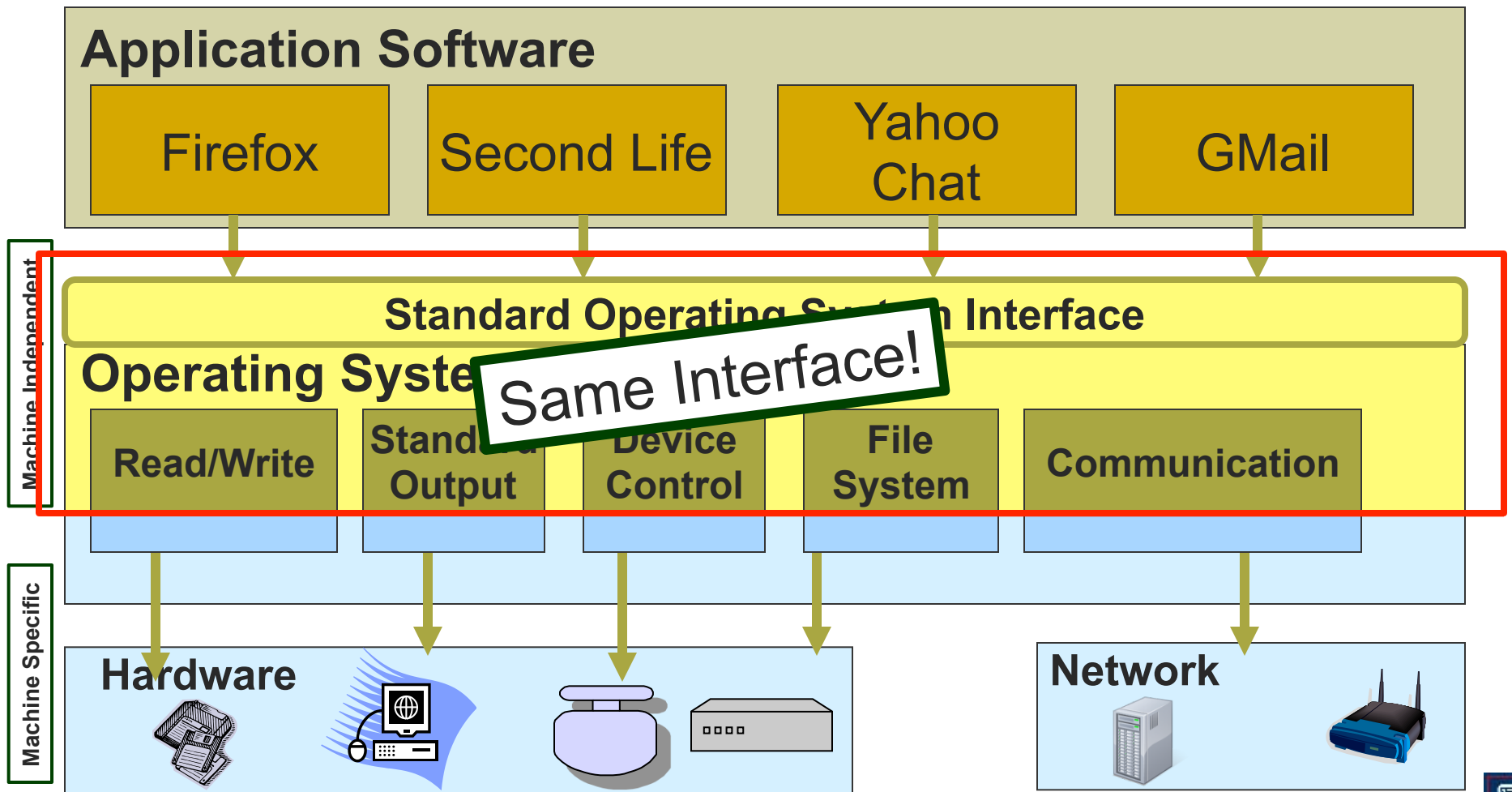
# Machine Independent = Portable



# OS Runs on Multiple Platforms

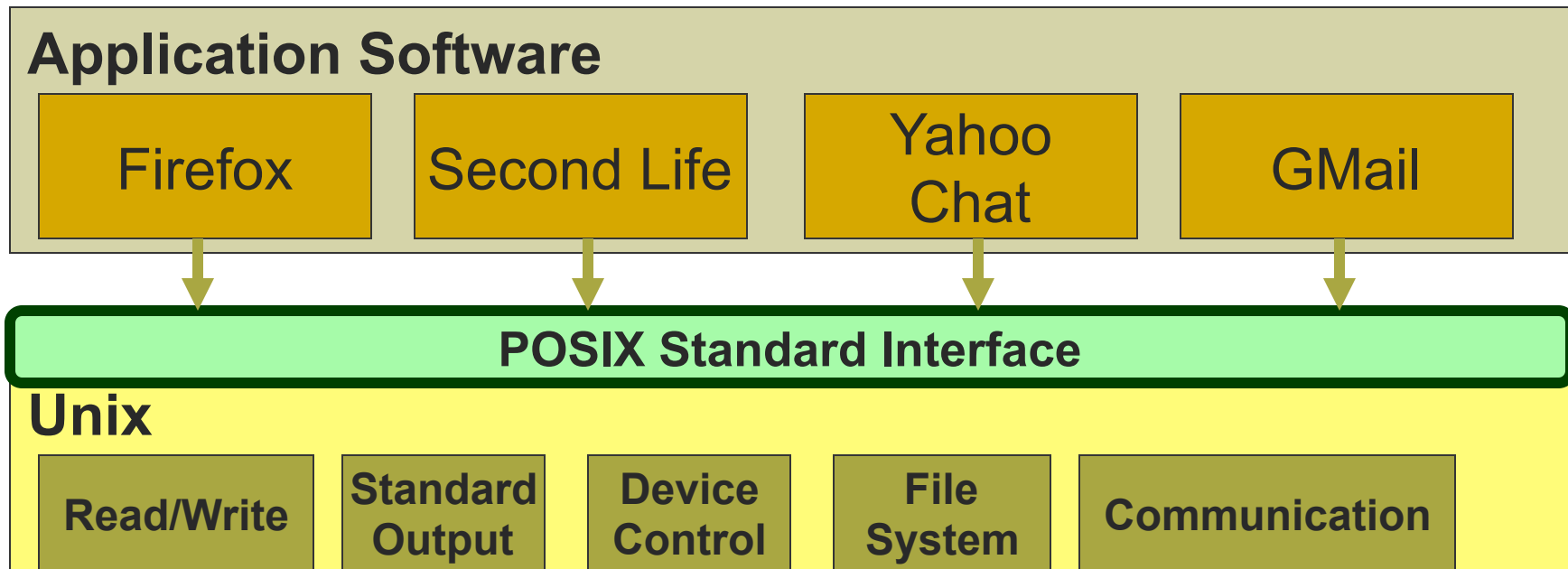


# OS Runs on Multiple Platforms



# 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 (tomorrow)
- <http://www.cs.illinois.edu/class/cs241>
  - Schedule is dynamic
  - Check regularly for updates
- Slides generally posted by morning of class
  - But some class material will not be in slides



# [ What comes next ]

- Switch out of 11am discussion section
- Visit the class webpage
  - Especially schedule, grading policy, homework & MP hand-in instructions, and resources
  - <http://www.cs.illinois.edu/class/cs241>
- Familiarize yourself with Piazza
- Refresh your C programming skills
  - <http://www.lysator.liu.se/c/bwk-tutor.html>
- Homework released tonight
- Next lecture: fun with C

