Welcome to CS 241
Systems Programming at Illinois

Marco Caccamo

&

Brighten Godfrey

The Team

- Marco Caccamo
  - Office: 4118 SC (office hours: Friday 11.50-12.50)
- Brighten Godfrey
  - Office: 3211 SC (office hours TBA)
- TAs
  - Zhongbo Chen, Rajath Subramanyam, Yang Xu, Fan Yang
- Discussion Sections
  - 8 sessions (Thursdays 9, 10, 12, 1, 2, 3, 4, 5)
  - All sections in SC 0220

News and Email

- Announcements and discussions: Piazza
  - https://piazza.com/class#spring2014/cs241
    - All class questions
    - This is your one-stop help-line!
    - Will get answer < 24 hours

- e-mail
  - cs241help-sp14@cs.illinois.edu
  - Only for personal questions not postable on Piazza

Are you trying to enroll, but cs241 is full?

- Stephen Herzog (smherzog@illinois.edu) handles the waiting list. Email him if you need to enroll. Hopefully, some space will open up within 1 week.
- Cs241 staff can NOT solve this problem.

The Textbook (optional)

- 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, 11-11:50am
  - Attend 1 discussion section per week
- Study posted class lectures (textbook optional)
  - Reading assignments posted on webpage
- Programming assignments (8) 45%
  - Longer MPs are worth a little more
- Midterm 25%
  - Monday, March 10th time: TBD
- Final 30%
  - Check university calendar

MPs submission policy and regrades

- Check the syllabus for details at:

https://courses.engr.illinois.edu/cs241/syllabus.html

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), **<u>all</u>** 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
  - Discussing how you worked through a particular problem

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 (they will be video recorded too: **link will be shared asap!**)
- Pay attention to the discussions
- Ask questions, and participate
- Do the exercises in class
- Start the assignment the day it is handed out, not the day it is due

What is systems programming?

What is a system?

sys·tem **Noun /ˈsistəm/**

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, …

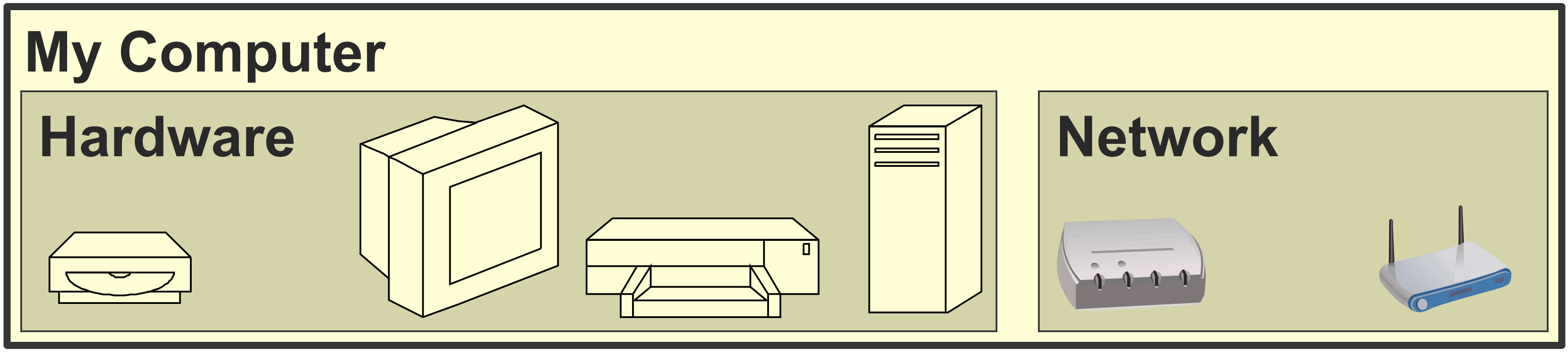
- Computer systems: a system of one or more connected computers and associated software
  - Search engines, social networks, databases, Internet
  - In this class, we learn how to design and code their software

Challenges in building computer systems

- Sharing resources among programs
- Preventing interference from malicious/ incorrect 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



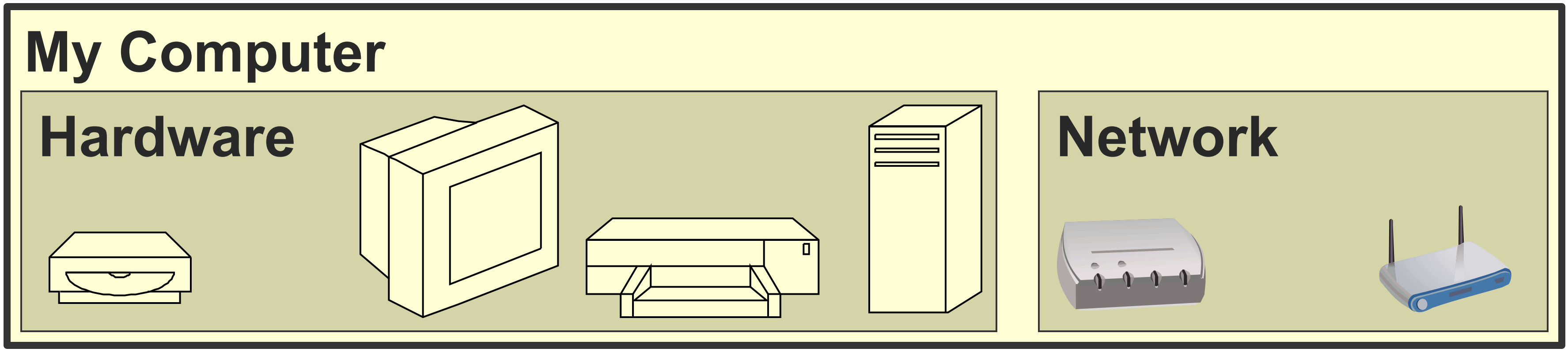
My Computer

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?



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

Application Software

Firefox

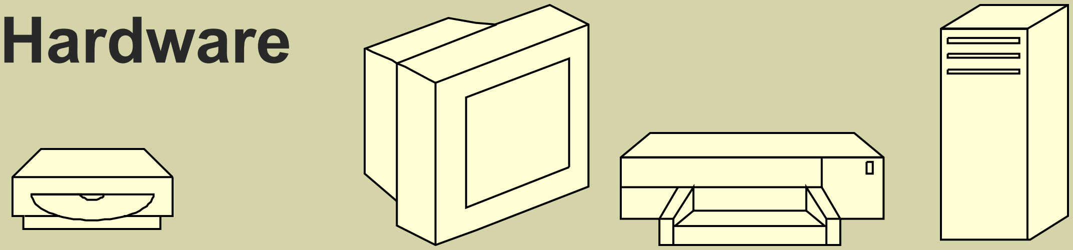
Second Life

Yahoo Chat

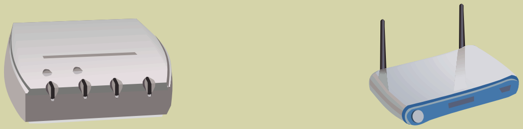
GMail

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

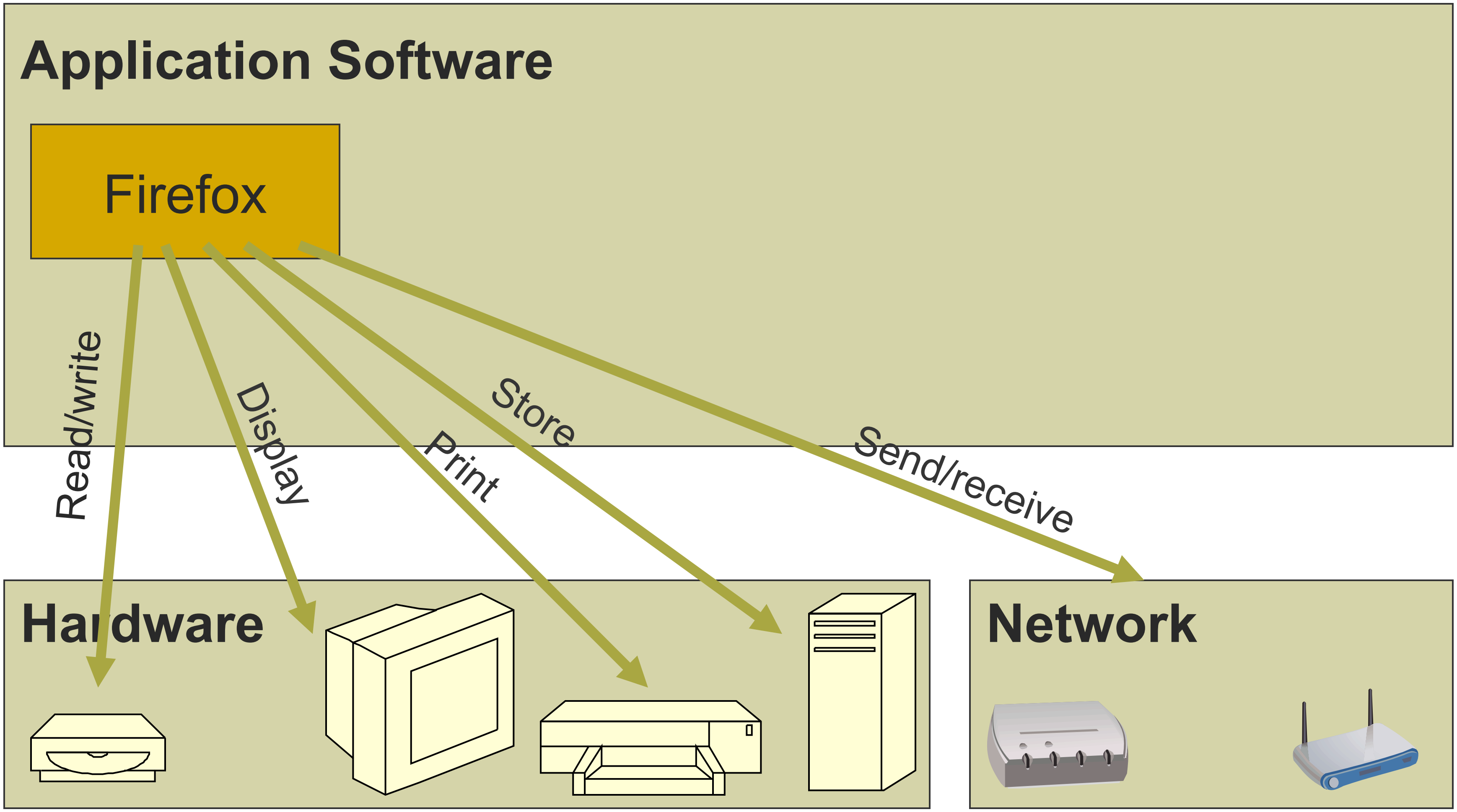
Hardware



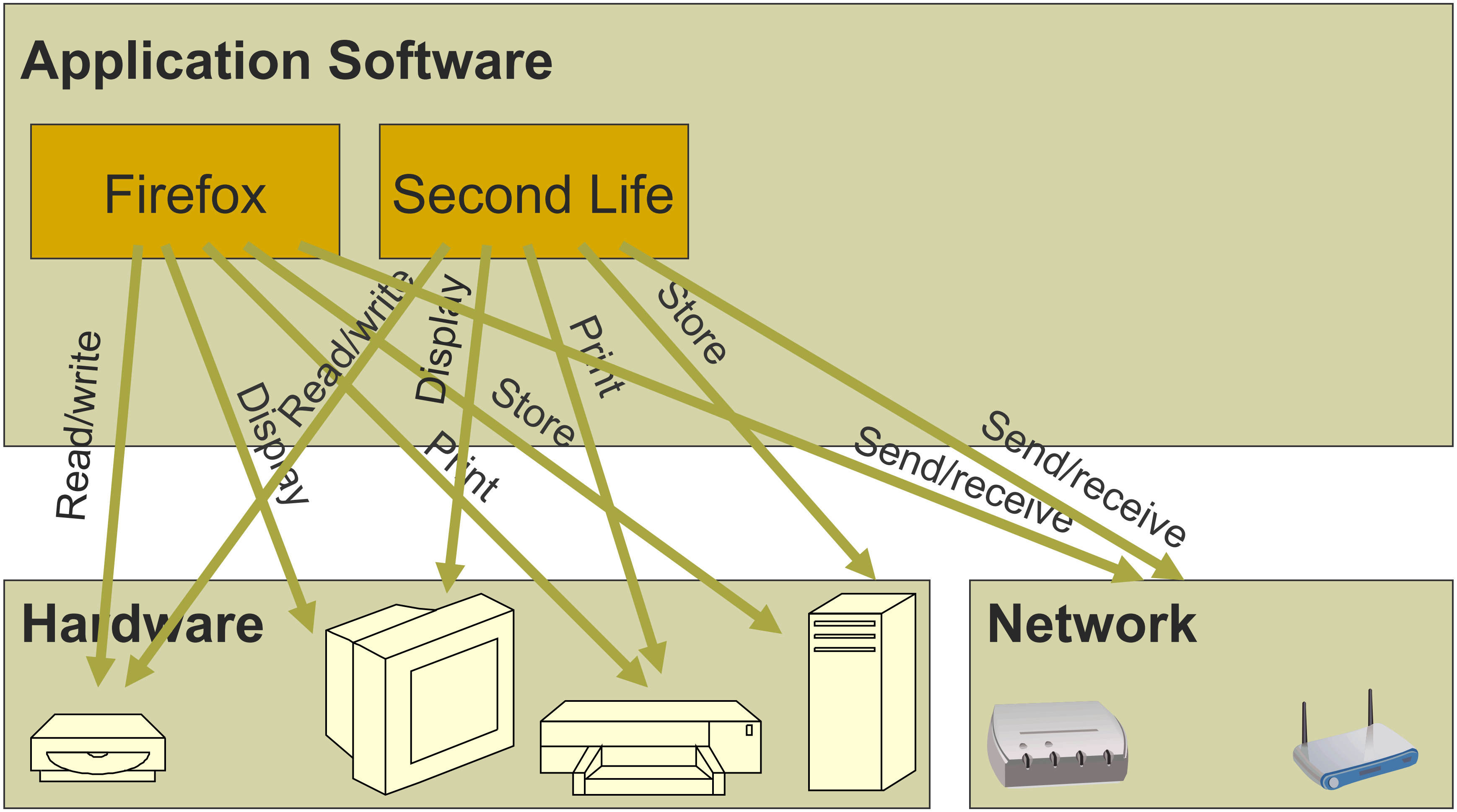
Network



Application Requirements

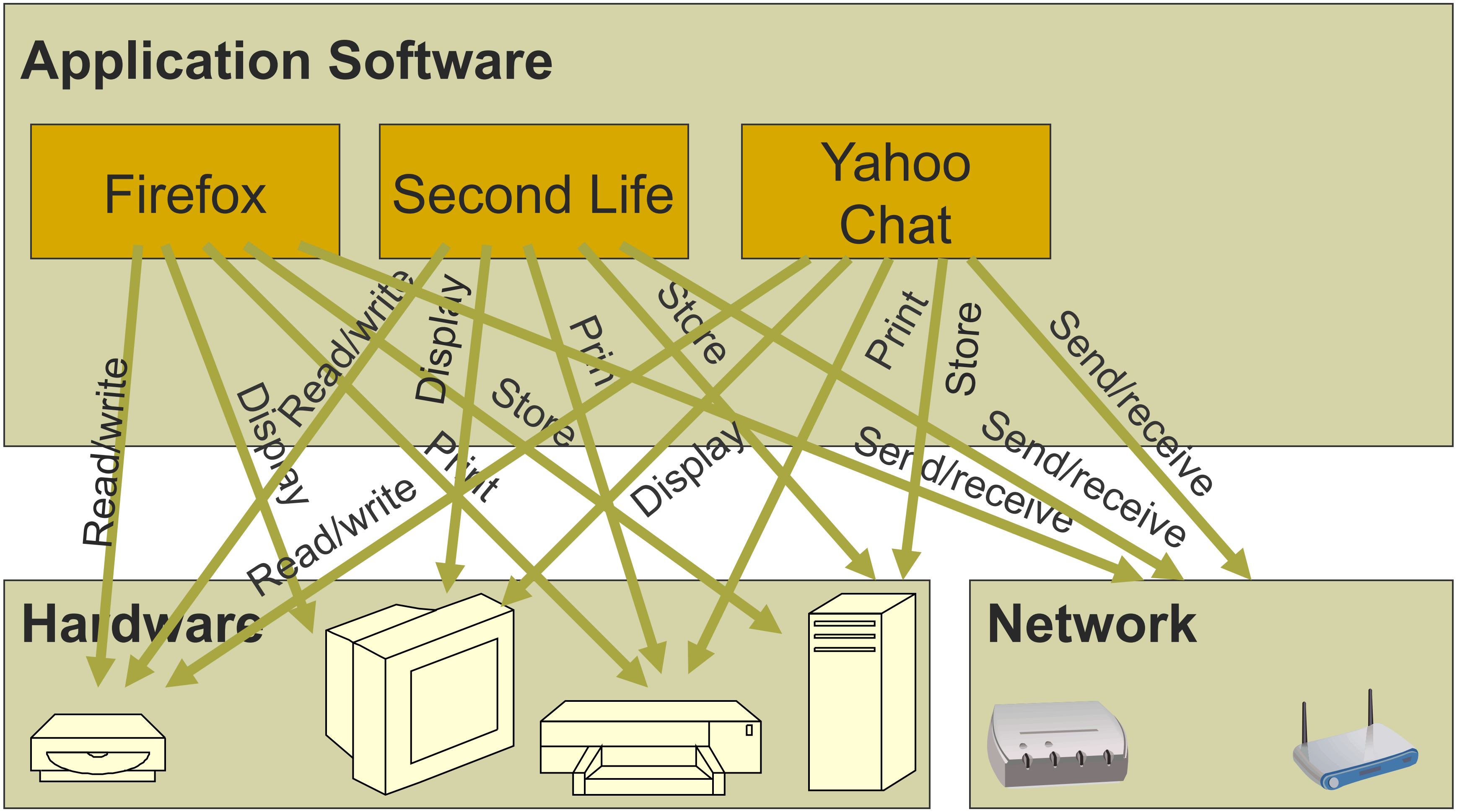


Two Applications?



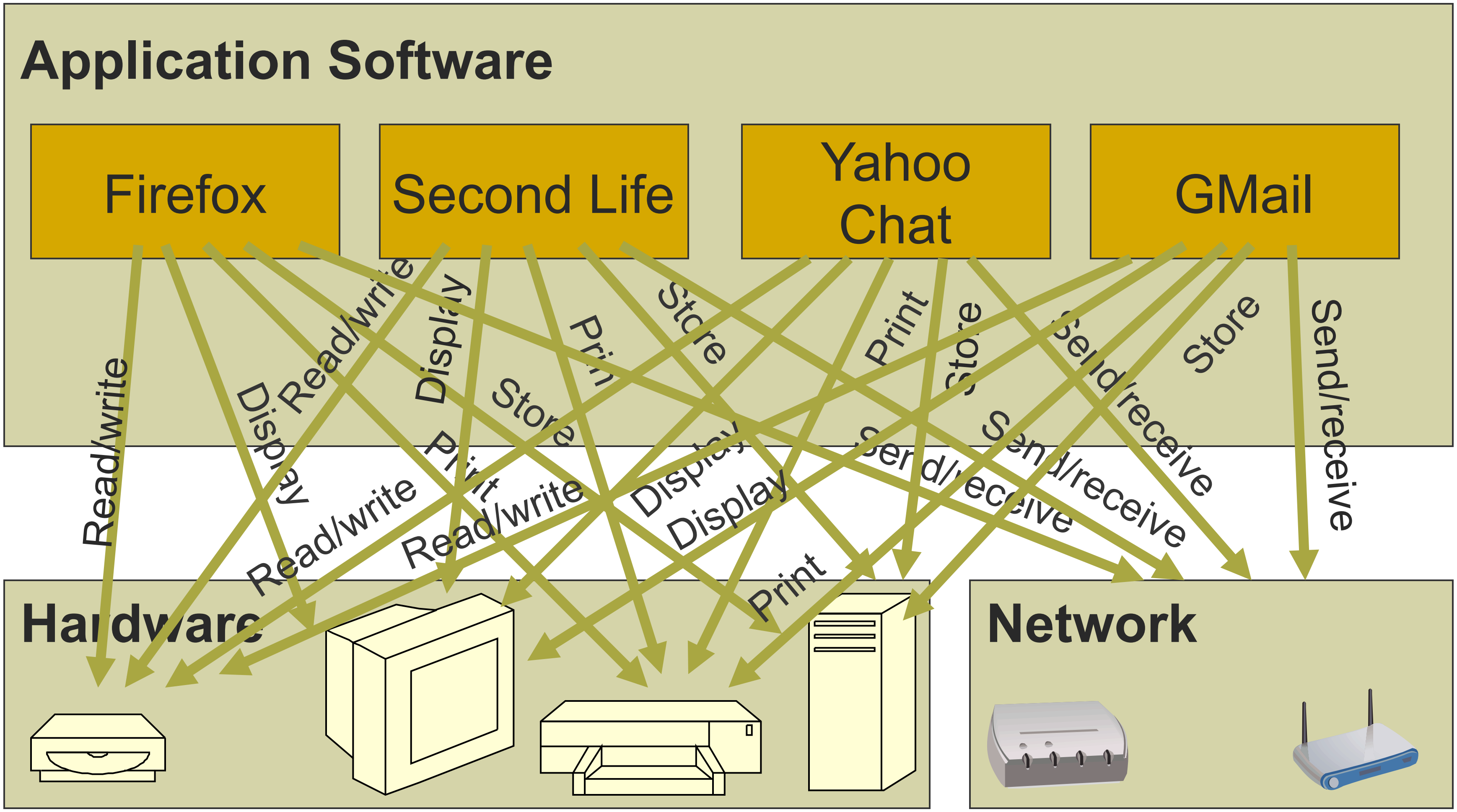
Copyright ©: University of Illinois CS 241 Staff

Managing More Applications?



Copyright ©: University of Illinois CS 241 Staff

We need help!



Application Software

Firefox

Second Life

Yahoo Chat

GMail

Read/write

Display

Read/write

Display

Print

Store

Store

Print

Store

Send/receive

Store

Send/receive

Read/write

Read/write

Display

Display

Print

Send/receive

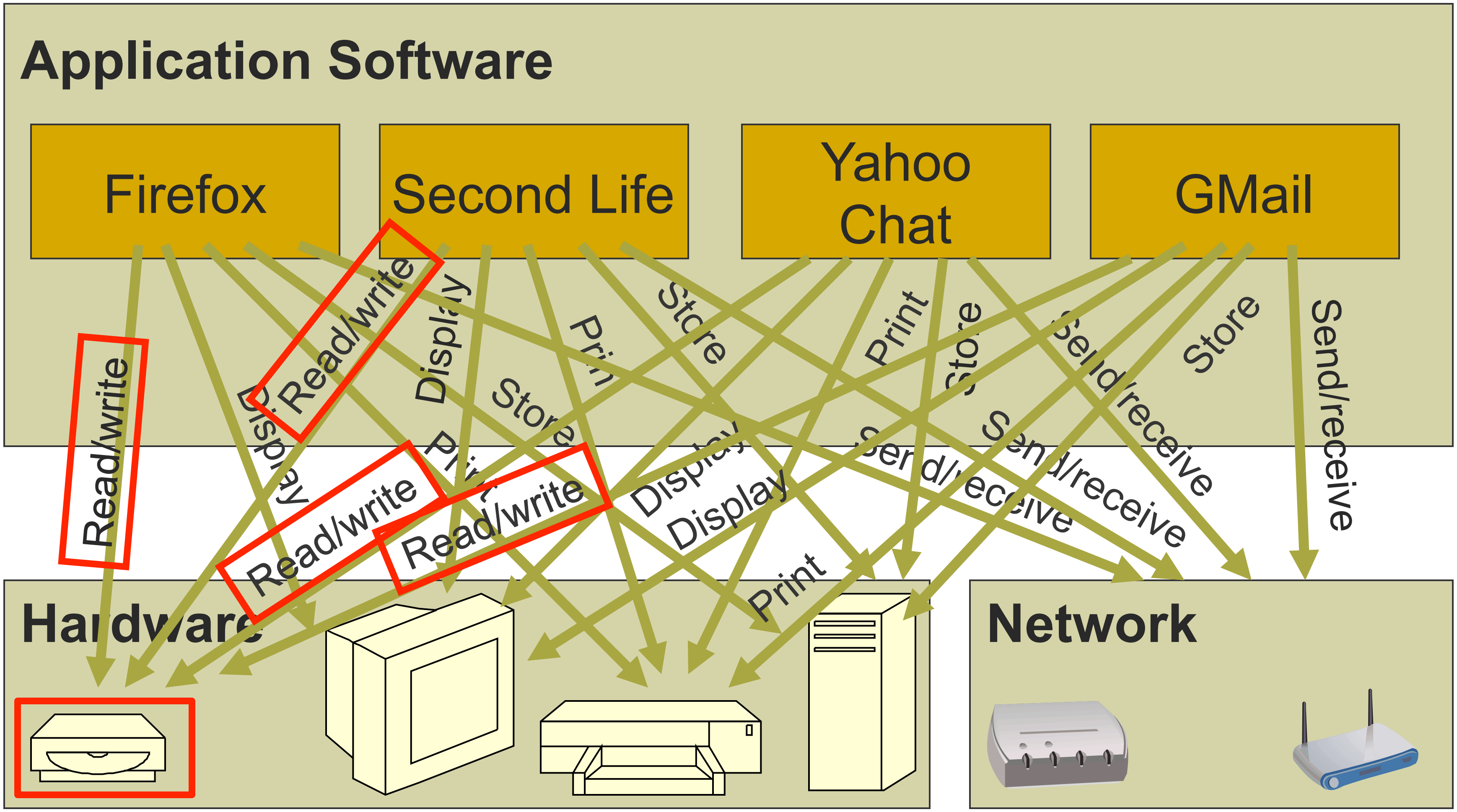
Send/receive

Print

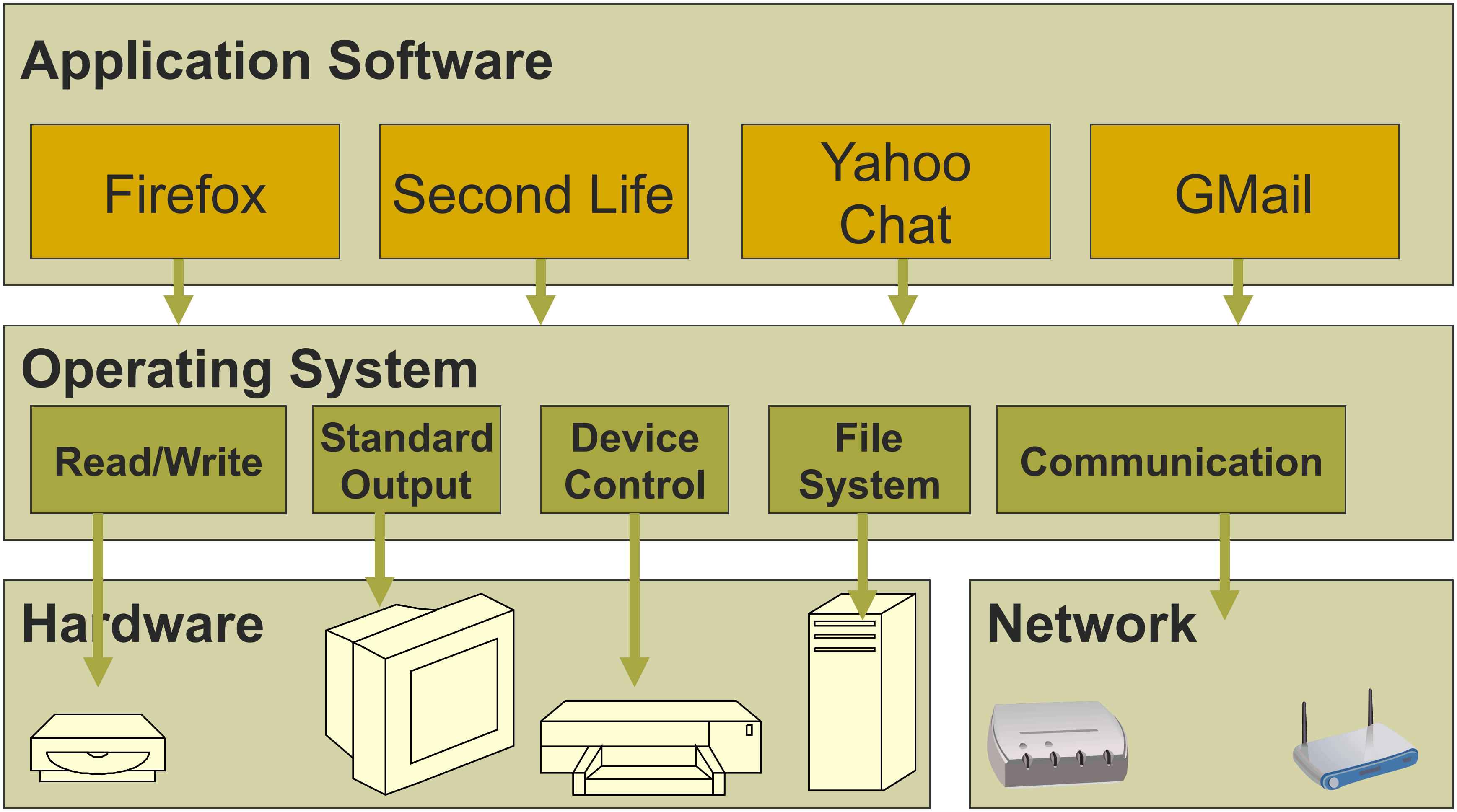
Hardware

Network

Approach: Find Common Functions



Delegate Common Functions



Application Software

Firefox

Second Life

Yahoo Chat

GMail

Operating System

Read/Write

Standard Output

Device Control

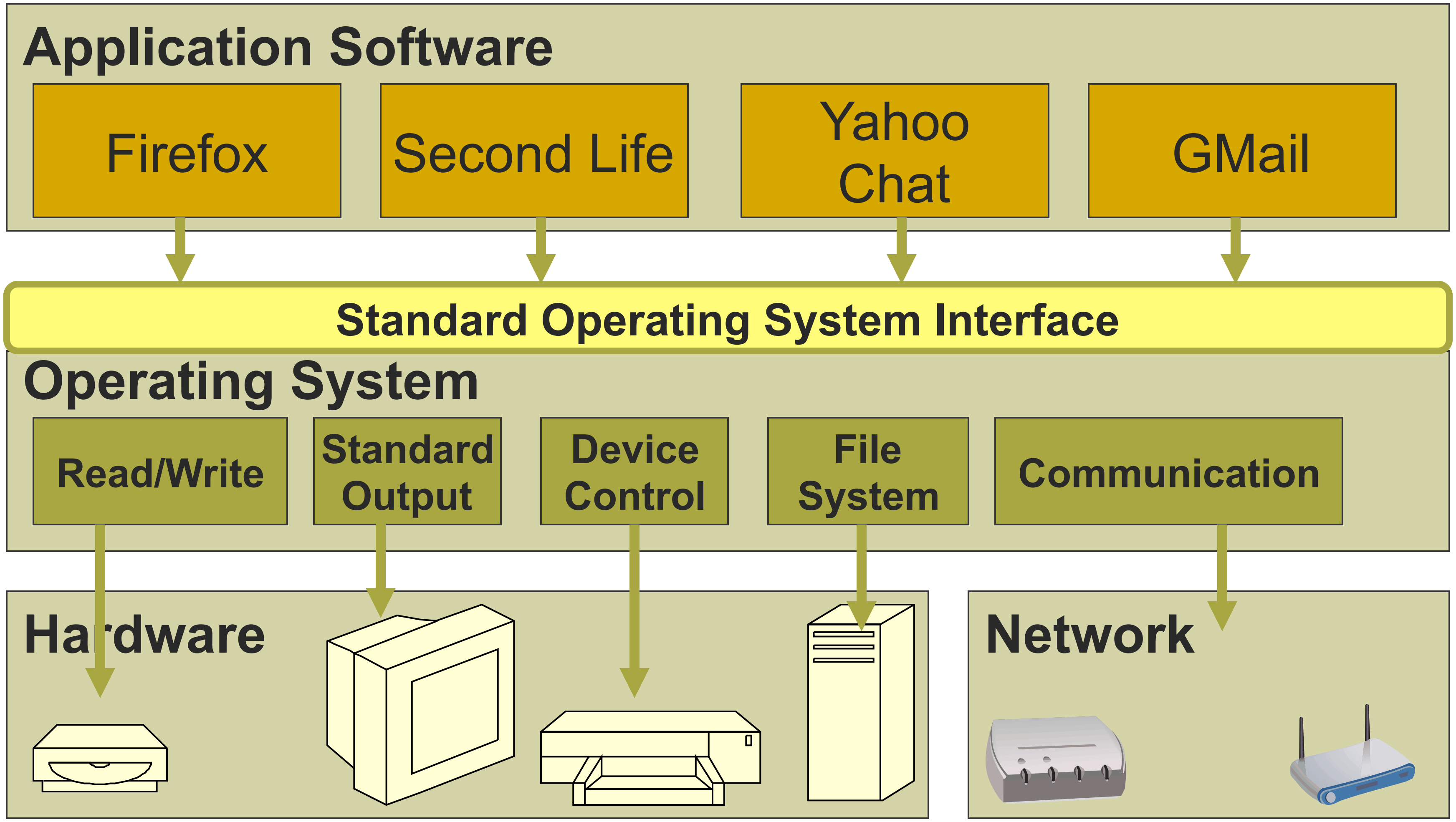
File System

Communication

Hardware

Network

Export a Standard Interface



Application Software

Firefox

Second Life

Yahoo Chat

GMail

Standard Operating System Interface

Operating System

Read/Write

Standard Output

Device Control

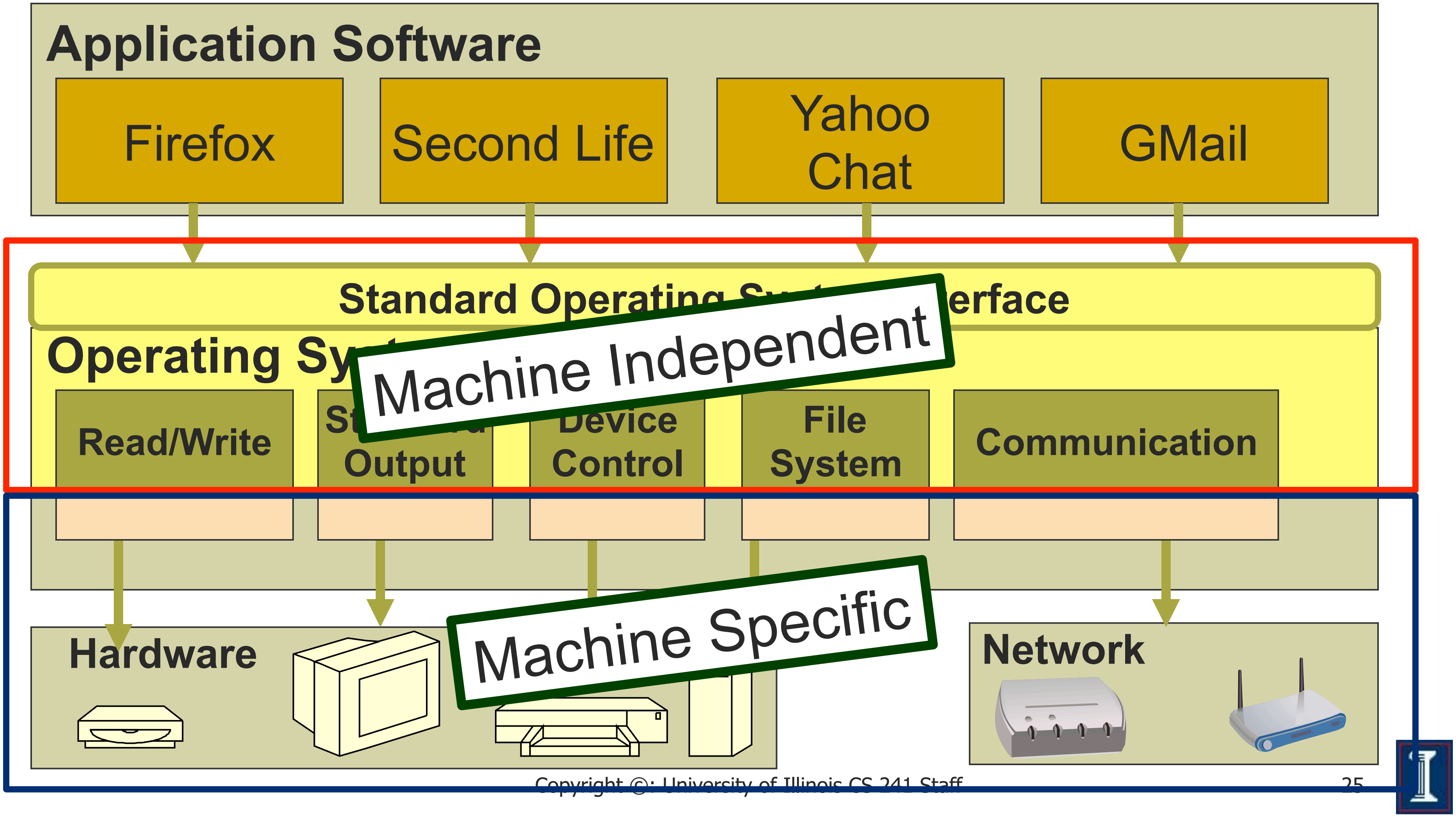
File System

Communication

Hardware

Network

Goal: Increase Portability



Application Software

Firefox

Second Life

Yahoo Chat

GMail

Standard Operating Sy erface

Operating Sy

Read/Write

St Output

Device Control

File System

Communication

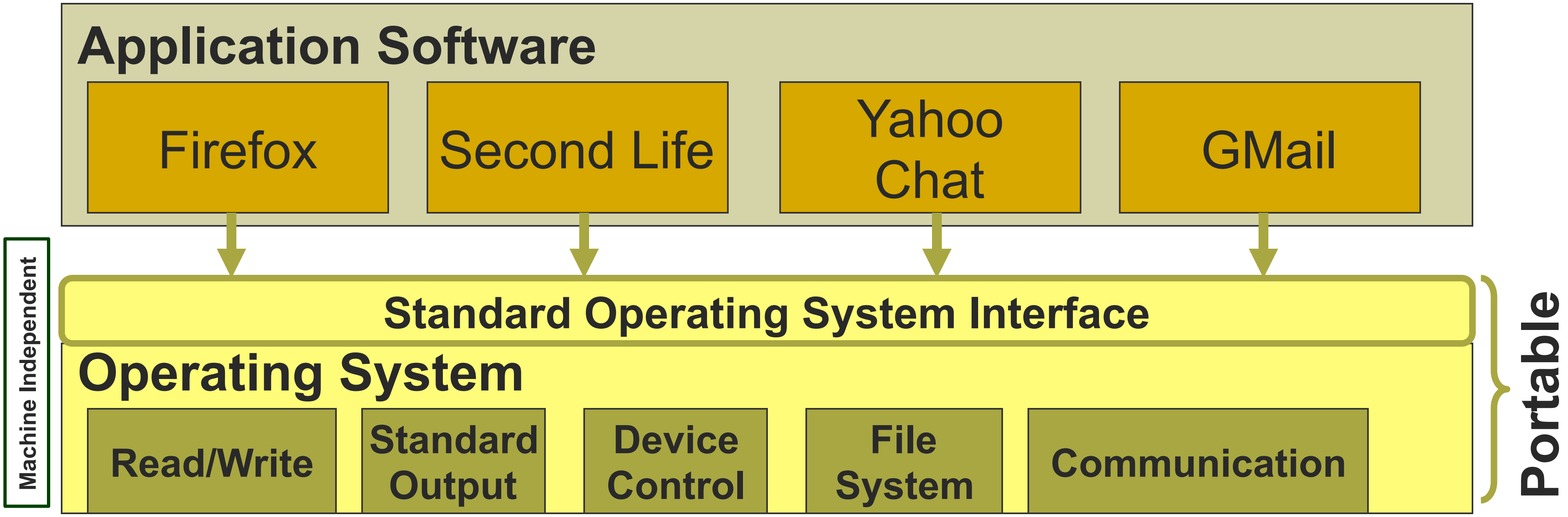
Machine Independent

Hardware

Machine Specific

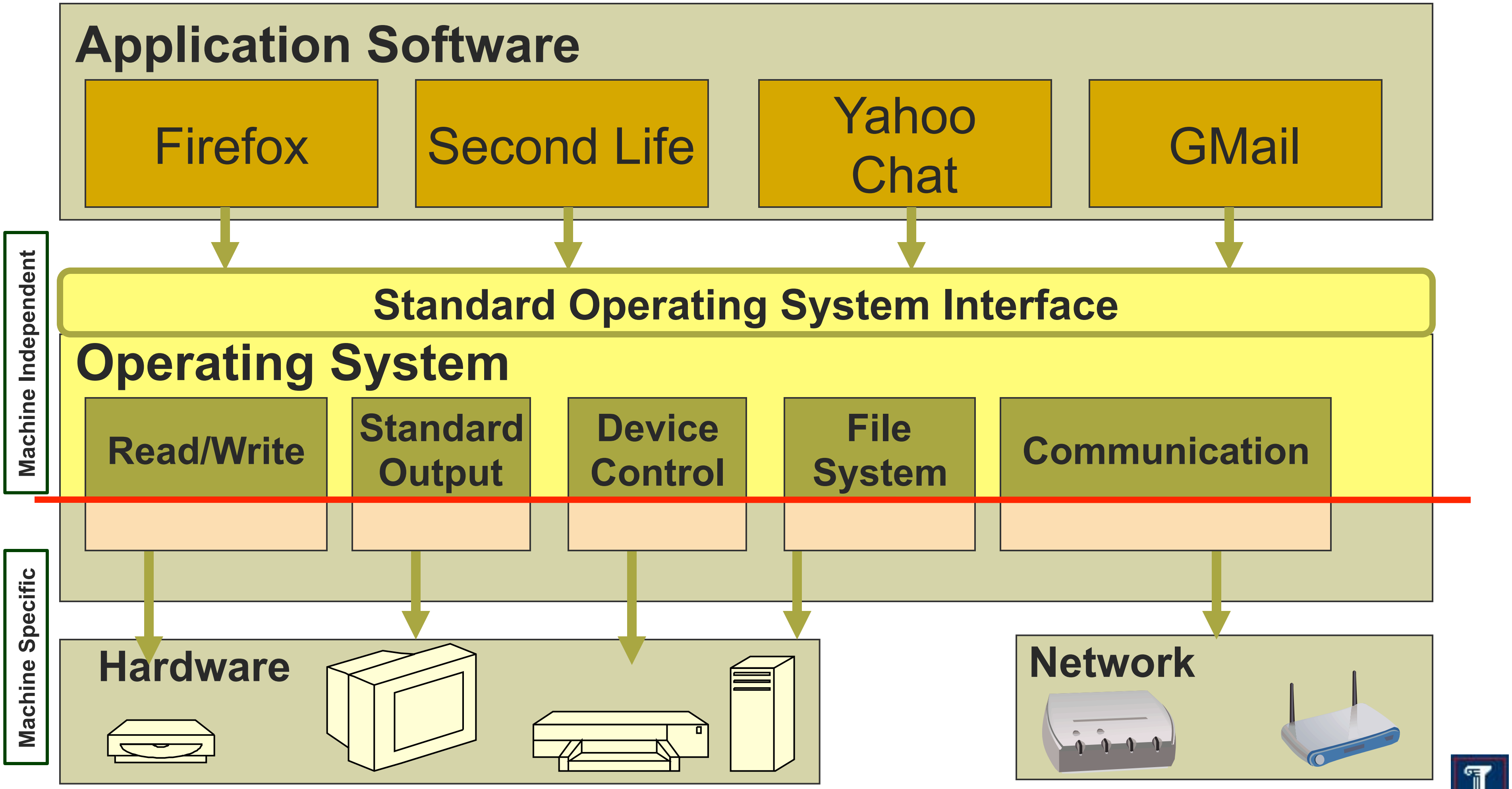
Network

Machine Independent = Portable

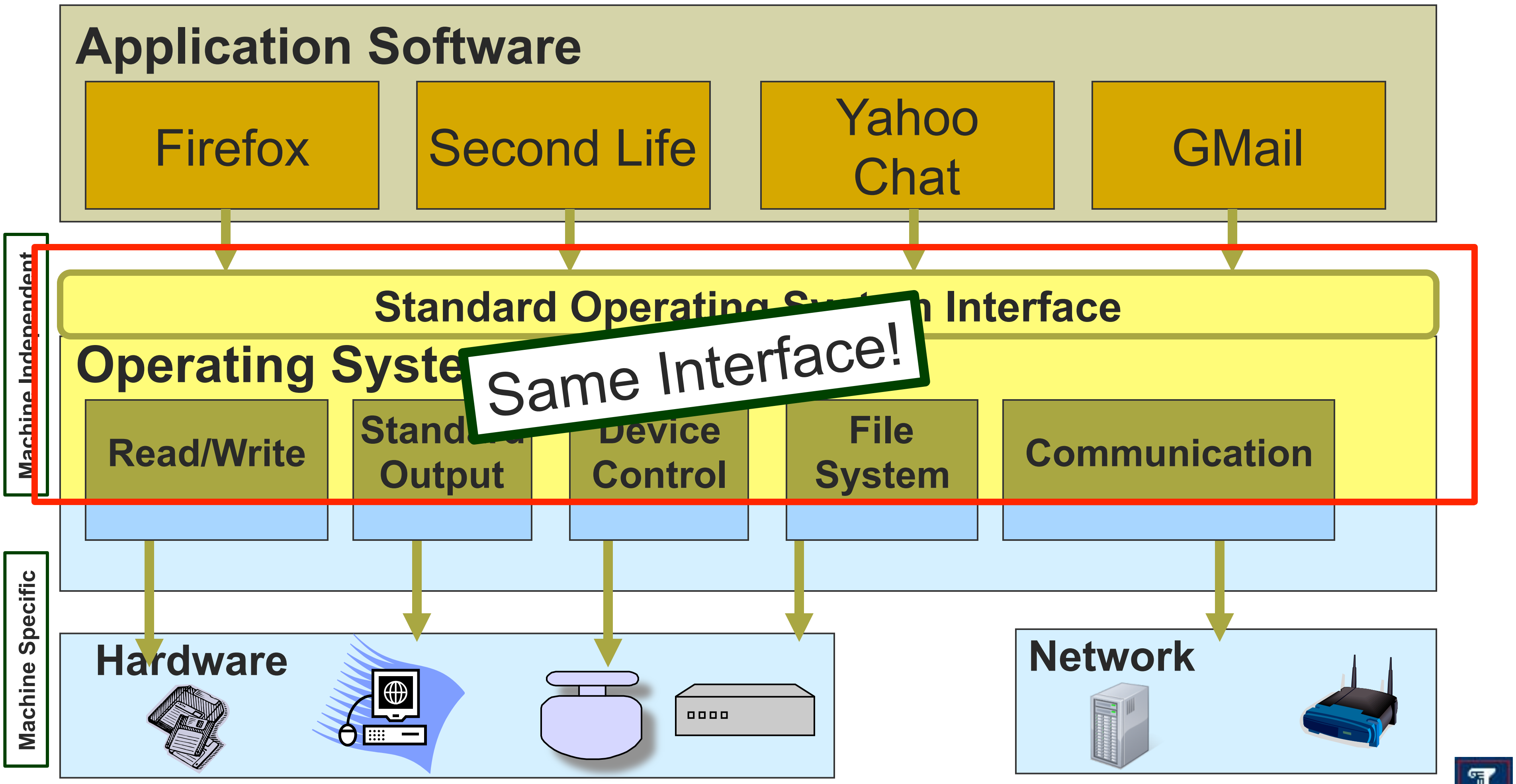


Copyright ©: University of Illinois CS 241 Staff

OS Runs on Multiple Platforms



OS Runs on Multiple Platforms



Application Software

Firefox

Second Life

Yahoo Chat

GMail

Machine Independent

Standard Operating Sy n Interface

Same Interface!

Operating Syste

Read/Write

Stand Output

Device Control

File System

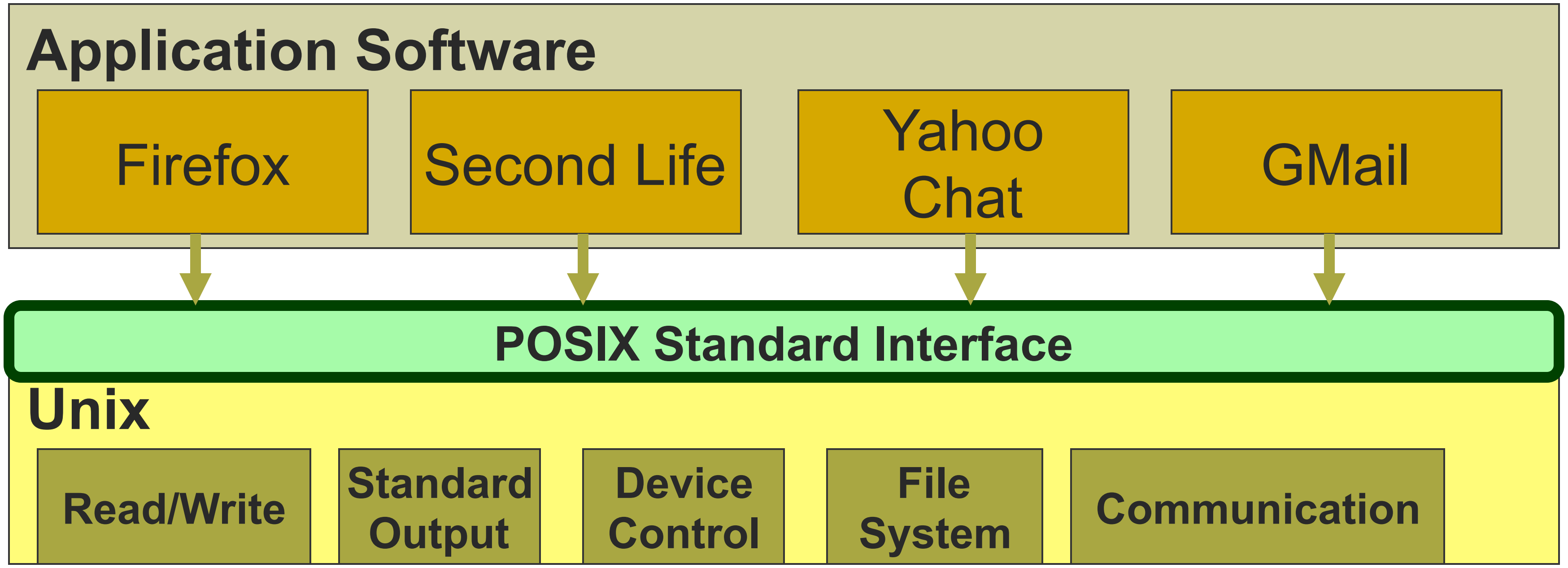
Communication

Machine Specific

Hardware

Network

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
  - MP0: Baby-steps in C (**to be released today!**)
  - 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
- Week 7-11: **Cooperating parallelism**
  - Communicating & sharing resources between proc./threads
  - MP5: Parallel make
  - MP6: 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
  - MP7: 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