Comp 533-Distributed Systems: Course Info

Instructor: Prasun Dewan (FB 150, dewan@unc.edu)

Distributed Program?

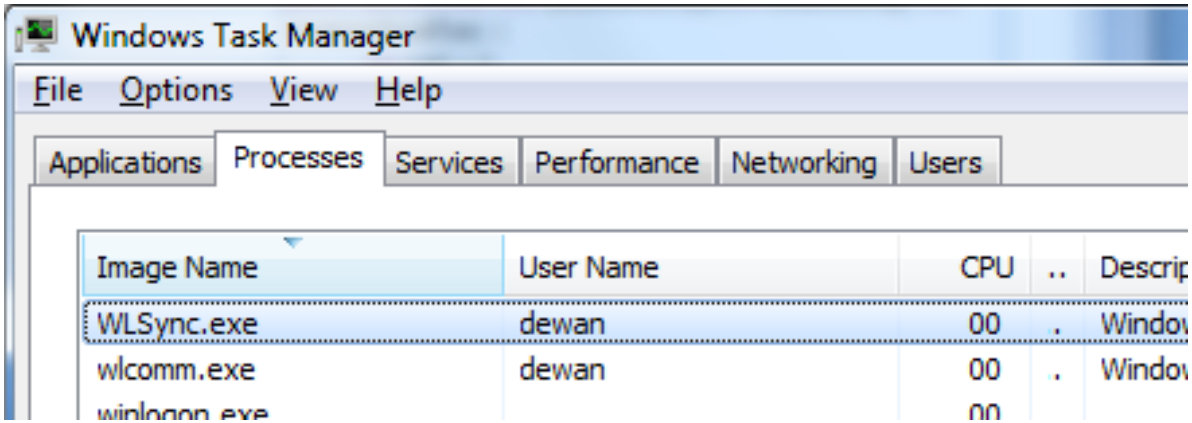
A program "involving" multiple computers

Specific computers must be bound at run time

→ Program can run on a single computer

Definition involves processes

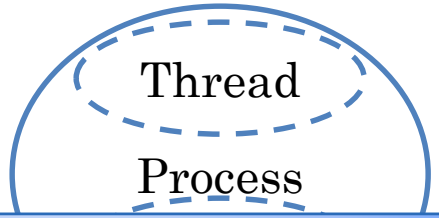
Program vs. Process vs. Thread



Program

```
public class AP2PAliceTOTSimulation {
    public static String ID = "9100";
    public static String NAME = "Alice";
    public static int USER_NUMBER = 0;
    public static void main (String[] args) {
        Tracer.showInfo(true);
```

Execution instance



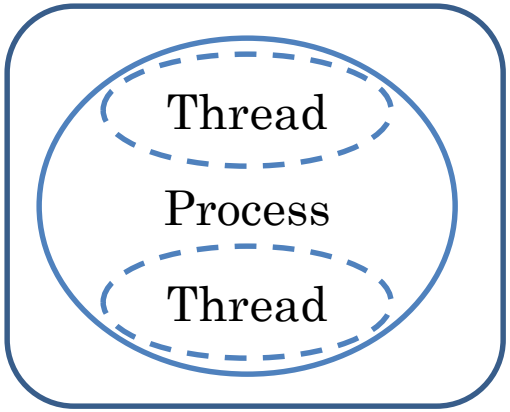
Process is execution instance of program, associated with program and memory

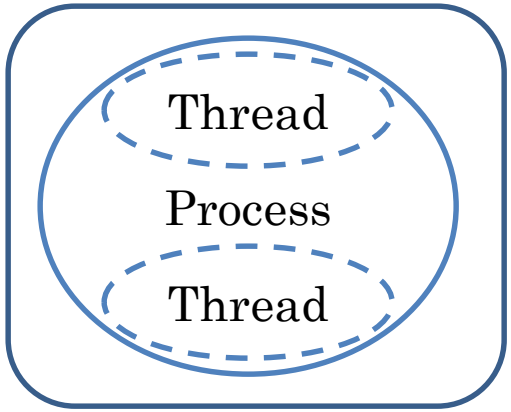
Processes are independent activities that can interleave or execute concurrently

Same program can result in multiple processes

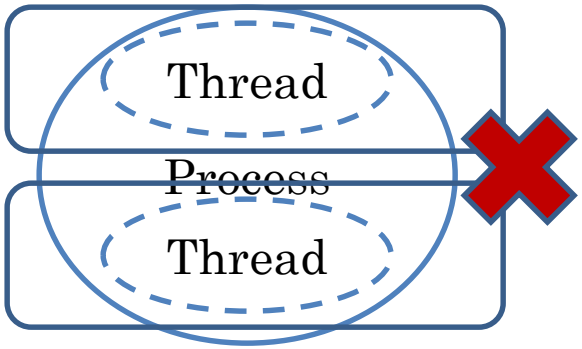
Thread is also an independent activity, but within a process, associated with a process and a stack

Distribution of Processes/Threads



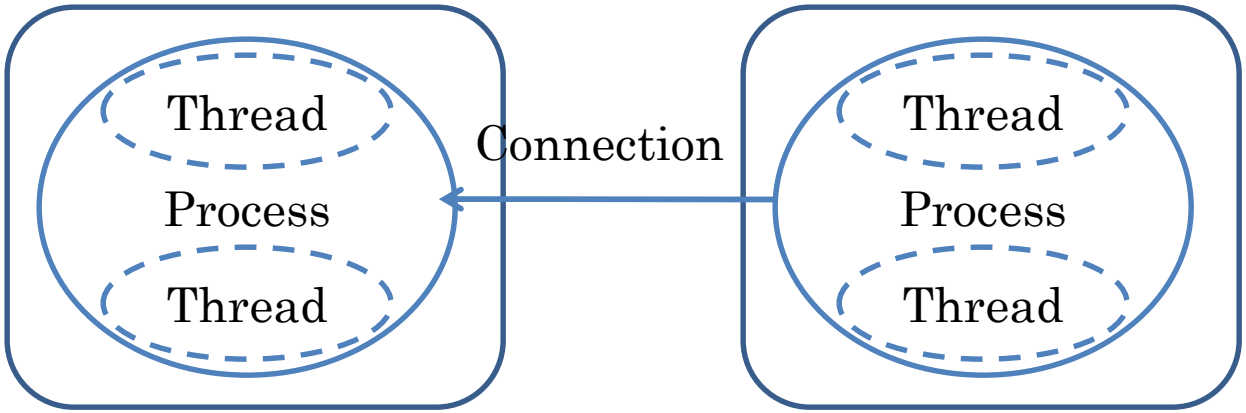


Different processes can execute on different (distributed) computers



A single process executes on one machine

Distributed Program



Exec

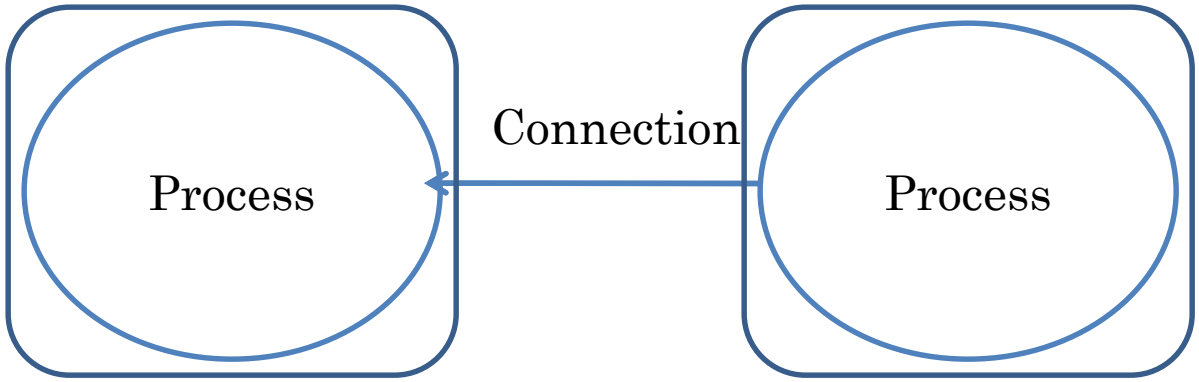
Connected process pair : Some computation of a process can be influenced by or influence computation of the other process

Connected process group: each process is coupled to at least one other process in the group

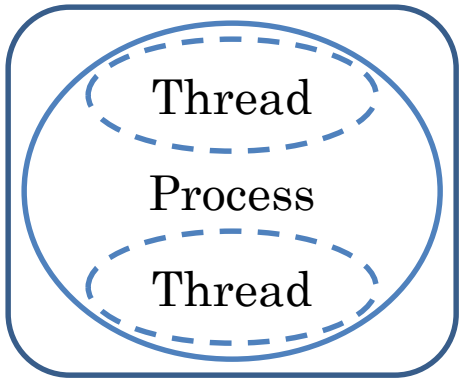
Graph created by creating pair-wise dependency links is not partitioned– every node reachable from every other node

```
public
    public static String ID = "9100";
    public static String NAME = "Alice";
    public static int USER_NUMBER = 0;
    public static void main (String[] args) {
        Tracer.showInfo(true);
        AP2PTOTSessionsClientCreator.createP2P
    }
}
```

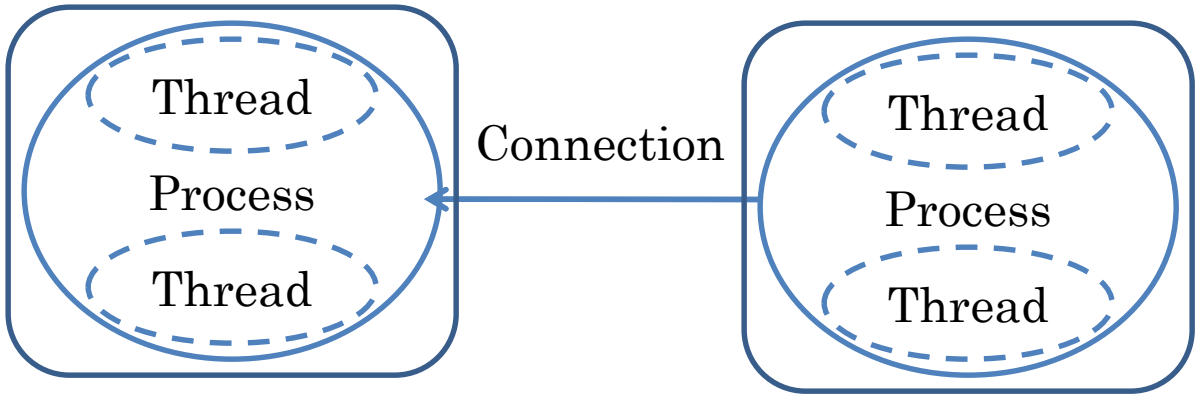
Distribution vs. Concurrency



Distribution, no fine-grained concurrency



Concurrency, not distribution



Distribution and fine-grained concurrency (typical)

Non-Distributed vs. Distributed Program

Non-Distributed	Distributed
Creates a single process unconnected to any other process	Creates a pair or larger group of connected processes
Must deal with sequential and possibly concurrency issues	Must also deal with distribution and usually concurrency issues

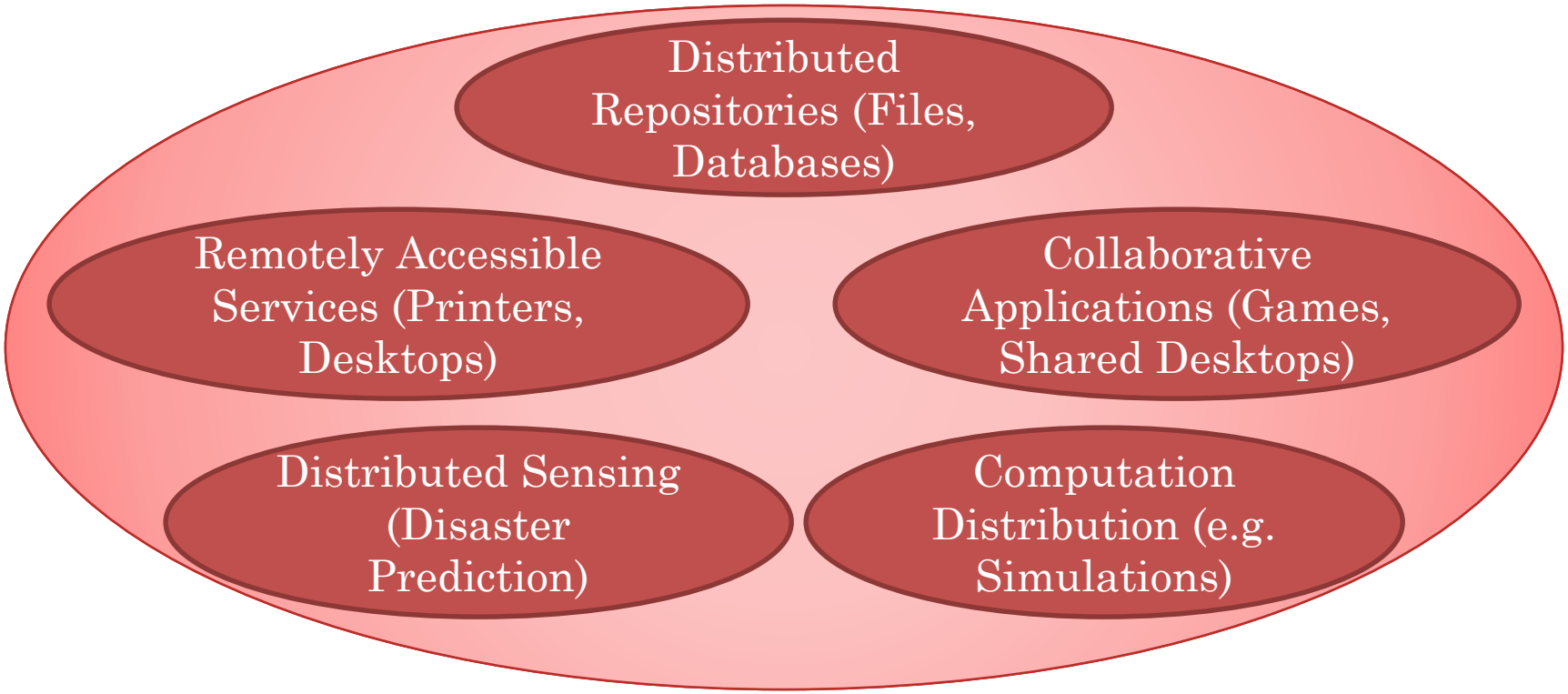
Examples of Practical Distributed Applications

Distributed applications?

Non distributed applications?

In today's world, what is or should not be distributed?

Examples



Will look at domain-independent, "foundational" concepts at the intersection of them

Non-Distributed vs. Distributed Program

Non-Distributed	Distributed
Creates a single process unconnected to any other process	Creates a pair or larger group of connected processes
Must deal with sequential and possibly concurrency issues	Must also deal with distribution and usually concurrency issues

Foundational concepts for non-distributed CS areas still relevant

How should these areas be adapted?

Classifying Foundational Courses

CS-1(Basic Programming Concepts)

CS-2(Program Design)

Data Structures: Implementation of Big Data Structures, small algorithms

Algorithms: Proof of small data Structures, Big Algorithms

Machine organization and assembly language

(Implementation of) Data Structures and Algorithms in OS, Compilers, Networks

Abstraction design

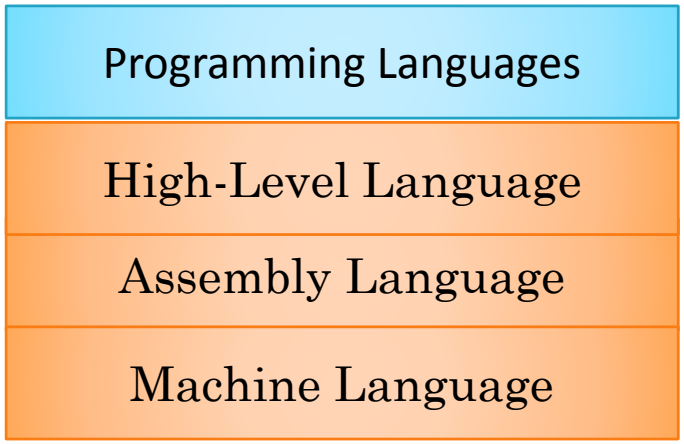
Abstraction use

Abstraction Implementation

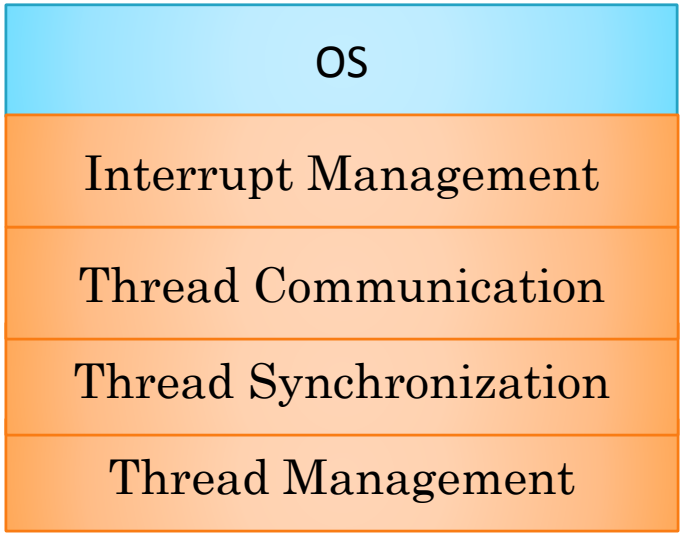
Implementation of Algorithms and Associated Data Structures

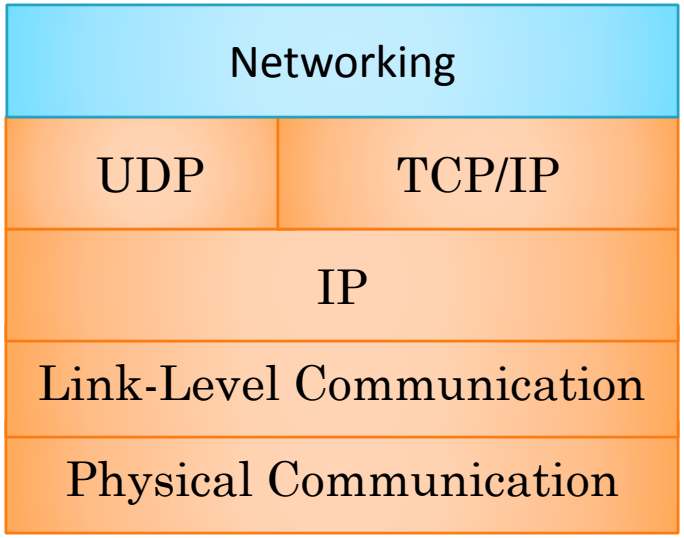
Proof of Algorithms and Associated Data Structures

Abstraction and Software Layering



Abstraction implementation involves mapping to lower-level abstraction(s)





Adding the Distributed Adjective

We will do an implementation-based coverage of selected concepts in design, use, and implementation of domain-independent distributed abstractions and algorithms (no text book)

Distributed operating systems covered in graduate courses and text books

Levels of abstraction and implementation?

Covered in Graduate course and several text books cover

Abstraction design

Abstraction use

Abstraction Implementation

Implementation of Algorithms and Associated Data Structures

Proof of Algorithms and Associated Data Structures

Networking vs. Distribution

Programming Languages
High-Level Language
Assembly Language
Machine Language

Communication in networking involves machines and uses hardware machine addresses

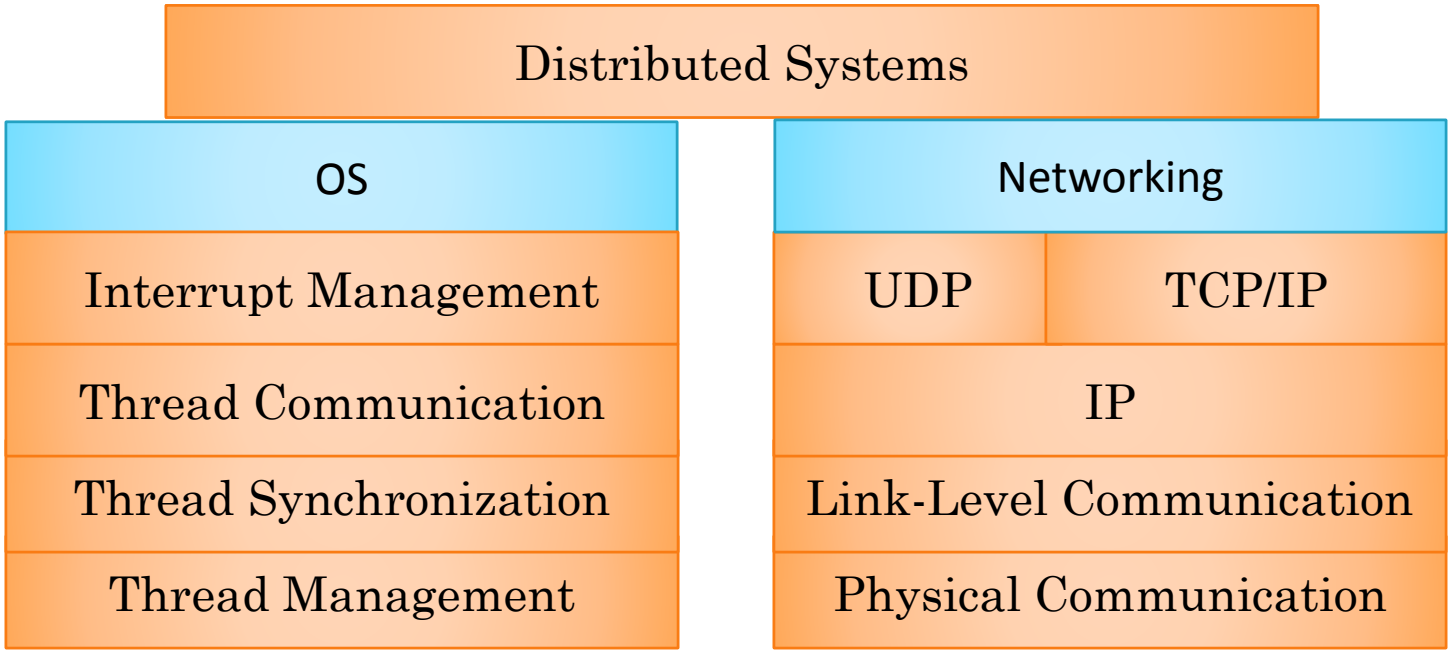
Communication in distributed systems is between processes and indicates routing of information among (OS) processes

OS
Interrupt Management
Thread Communication
Thread Synchronization
Thread Management

Networking	
UDP	TCP/IP
IP	
Link-Level Communication	
Physical Communication	

Base Layers for Distributed Systems

Layering in Distributed Systems?



Some Important Distributed Abstractions

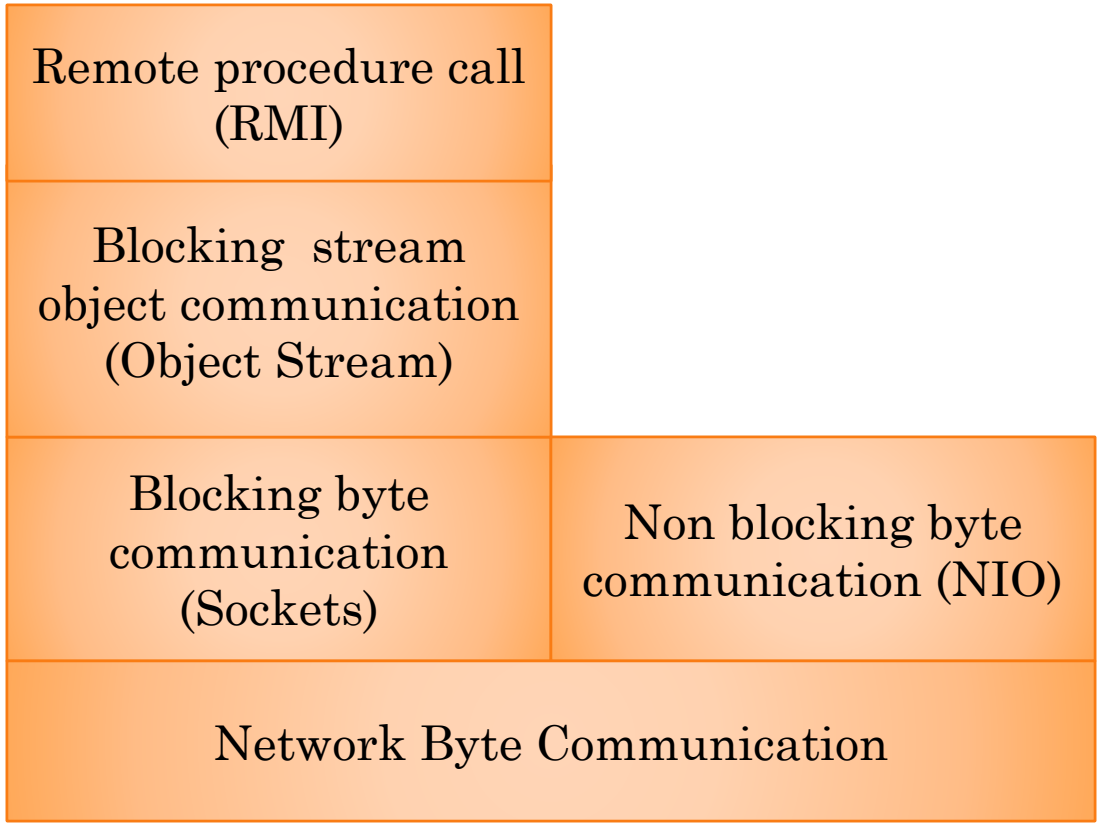
Blocking stream object communication (Object Stream)

Blocking byte communication (Sockets)

Non blocking byte communication (NIO)

Remote procedure call (RMI)

Distributed System Layers



Adding the Distributed Adjective: Scope of Course

- Design and use of (Java) NIO
- Design and use of (Java) RPCs (RMI and GIPC)
- Implementation of (Java) RPC in GIPC
- Design Space of Interprocess Communication
- Implementation of Algorithms for Fault-Tolerant Distributed Consensus in Replicated Objects

- Abstraction design
- Abstraction use
- Abstraction Implementation
- Implementation of Algorithms and Associated Data Structures
- Proof of Algorithms and Associated Data Structures

Non-Distributed vs. Distributed

Programming: Abstraction use

Programming: Use of a specific set of non distributed abstractions (e.g. , functional, MATLAB programming)

Distributed Programming : Use of a set of distributed abstractions (NIO, RMI, Replicated Objects with Consensus)

Systems: Abstraction design and/or implementation

Design and implementation of non distribution abstractions (Object-Oriented vs. Functional Languages, Compilers/Interpreters)

Design and implementation of distributed system abstractions (Serialization and RPC design and implementation)

Theory: Models and algorithms

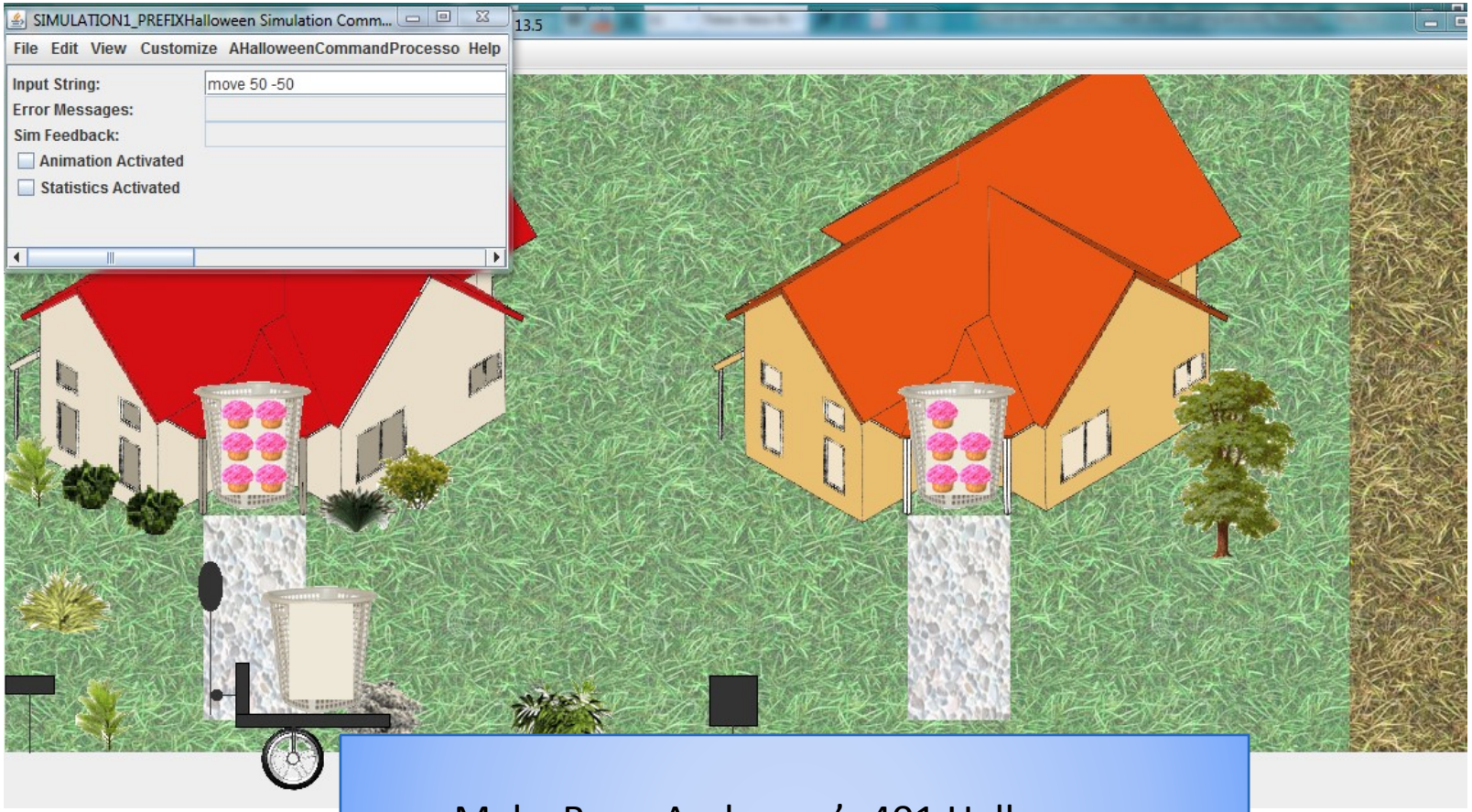
Non distributed model and algorithms (Turing Machines, HeapSort,)

Distributed Models and Algorithms (Fault Tolerant Consensus)

Implementation/Assignment-Based Course

Goal is to teach concepts that allow implementation of a set of course assignments

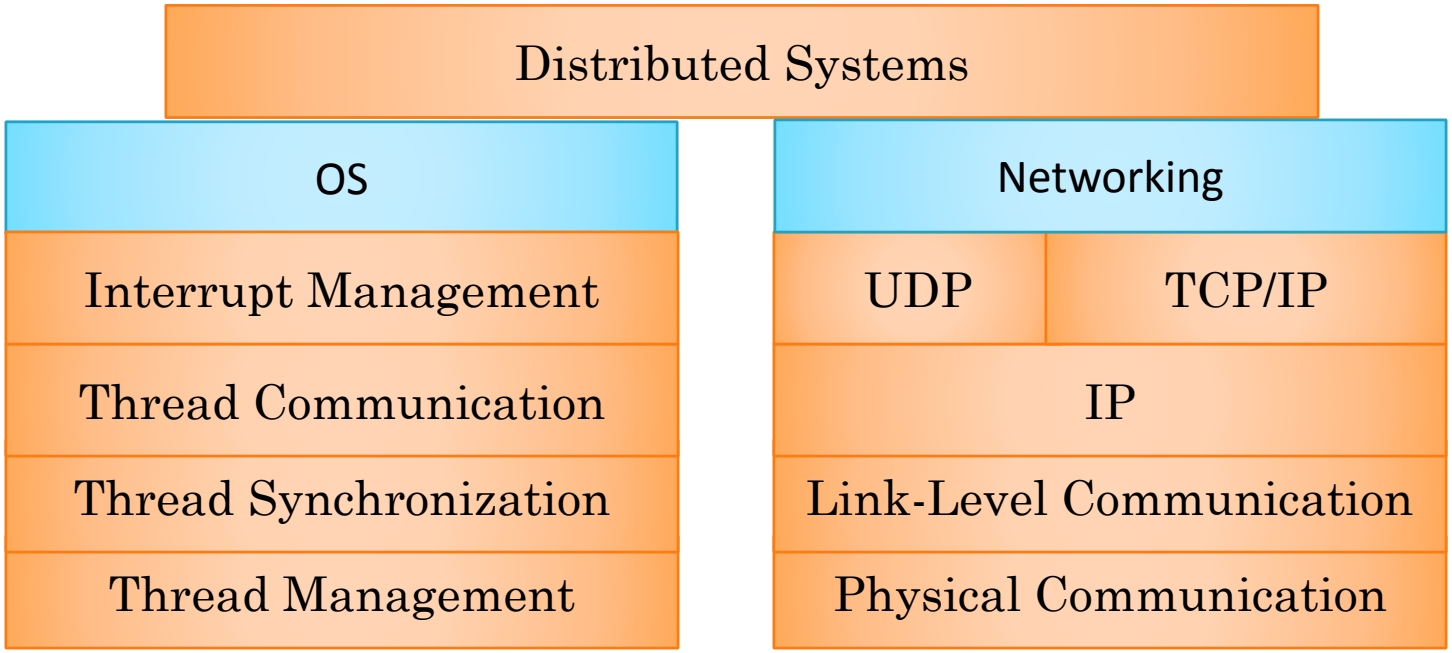
Assignments Nature: Halloween Simulation



Make Beau Anderson's 401 Halloween implementation distributed in multiple ways

Base Layers for Distributed Systems (Review)

Layering in Distributed Systems?



Implementation/Assignment-Based Course

Goal is to teach concepts that allow implementation of a set of course assignments

Adding the Distributed Adjective

We will do an implementation-based coverage of selected concepts in design, use, and implementation of domain-independent distributed abstractions and algorithms (no text book)

Distributed operating systems covered in graduate courses and text books

Levels of abstraction and implementation?

Covered in Graduate course and several text books cover

Abstraction design

Abstraction use

Abstraction Implementation

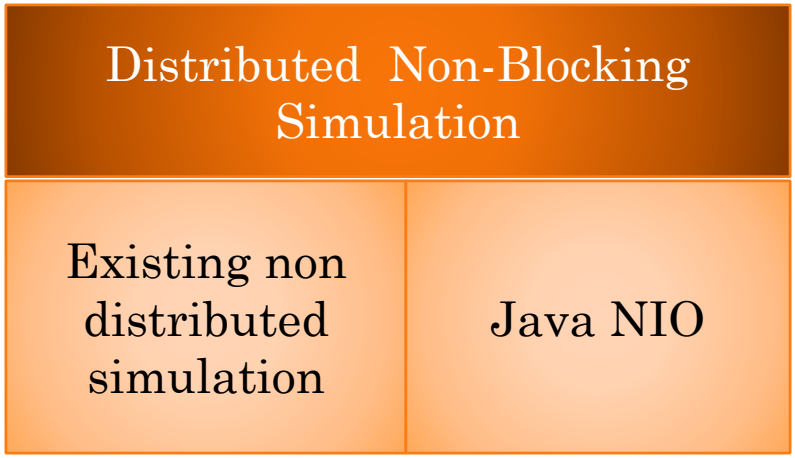
Implementation of Algorithms and Associated Data Structures

Proof of Algorithms and Associated Data Structures

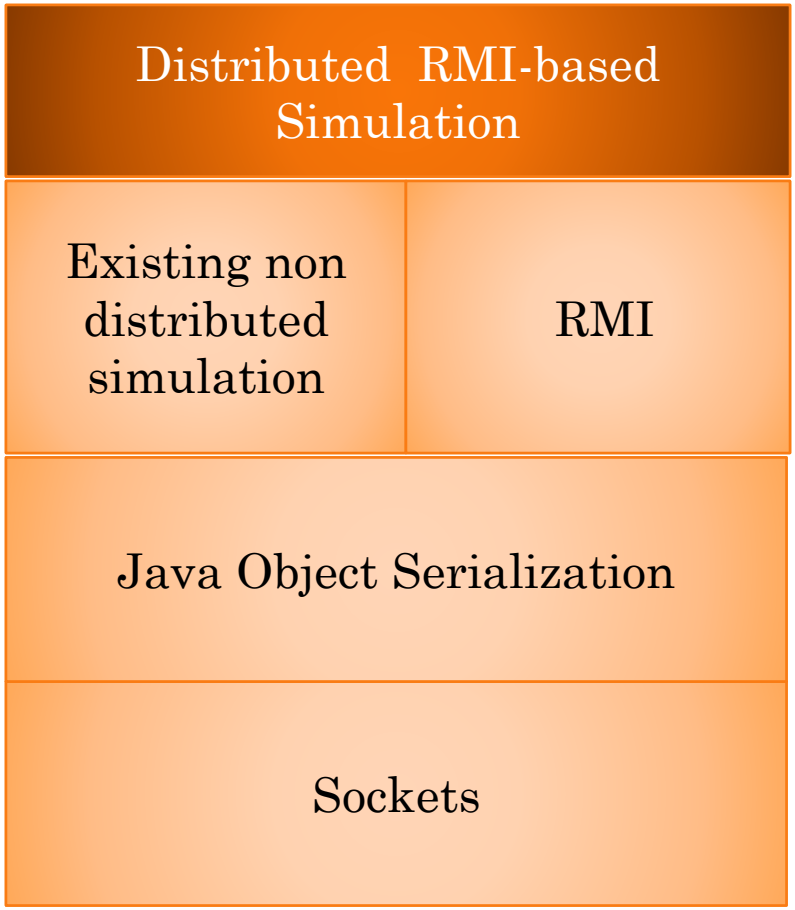
Assignments Nature: Halloween Simulation

Make Beau Anderson's 401 Halloween implementation distributed in multiple ways

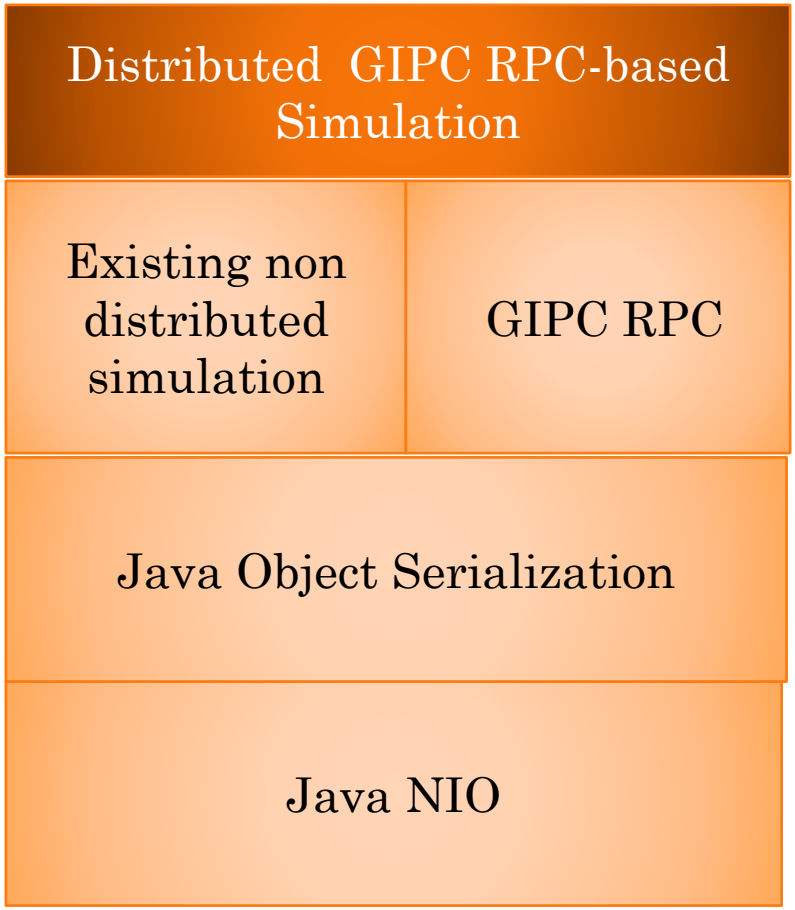
Assignment 1: Use Non Blocking I/O



Assignment 2: Use Synchronous RMI



Assignment 3-Part 1: Use Asynchronous RMI

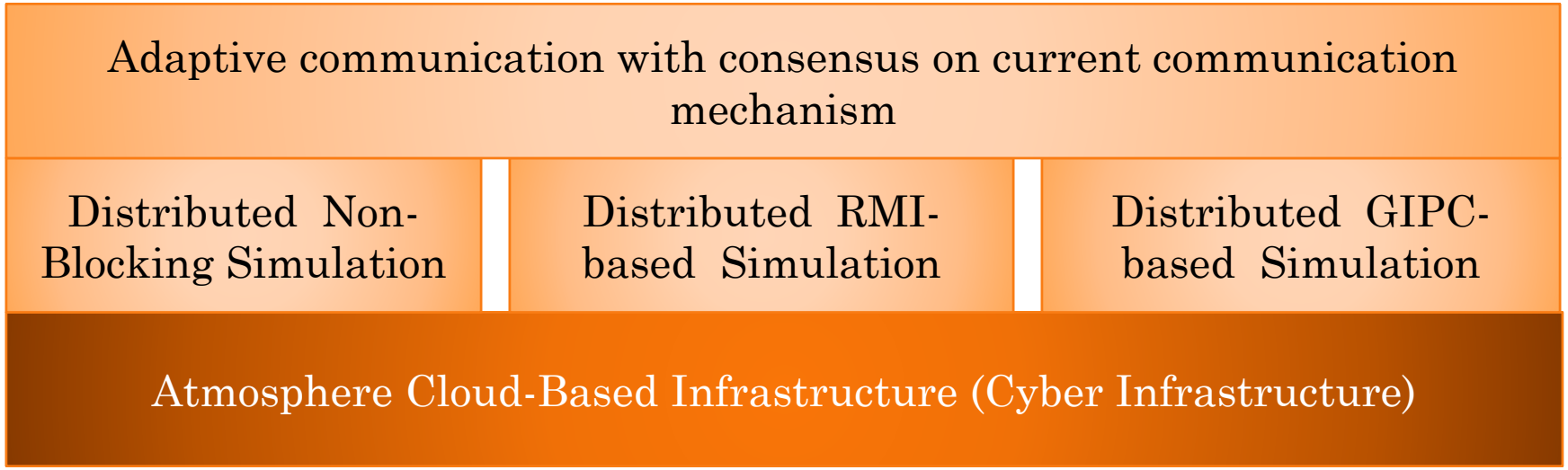


Assignment 3-Part 2: Centralized, Non-Fault Tolerant Consensus Algorithm

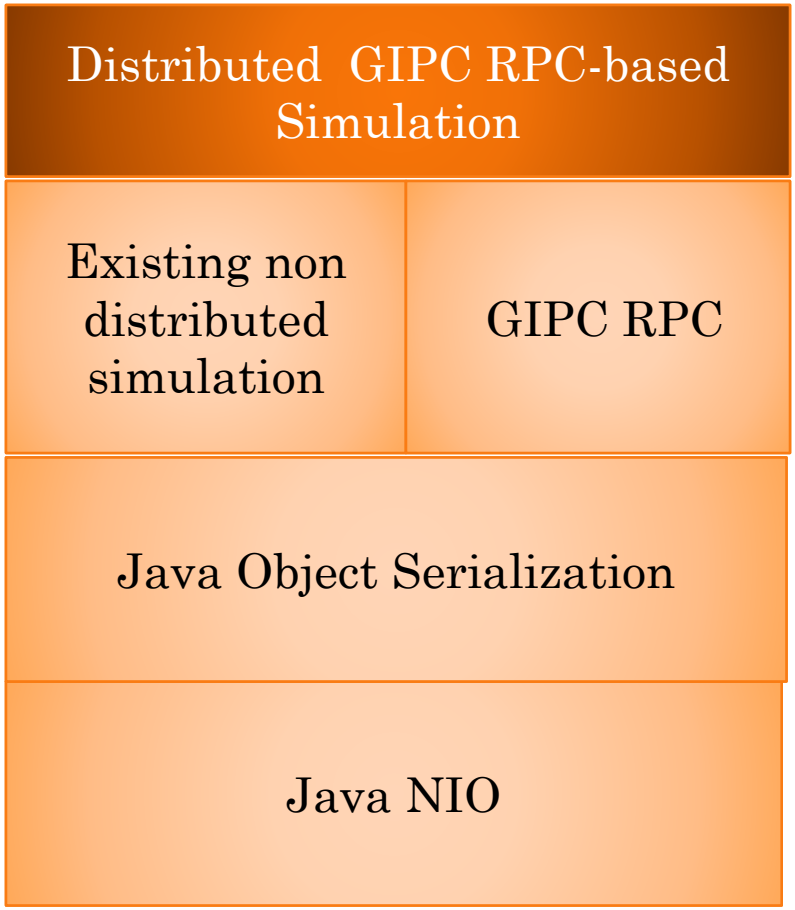
Adaptive communication with consensus on current communication mechanism

Distributed Non-Blocking Simulation	Distributed RMI-based Simulation	Distributed GIPC-based Simulation

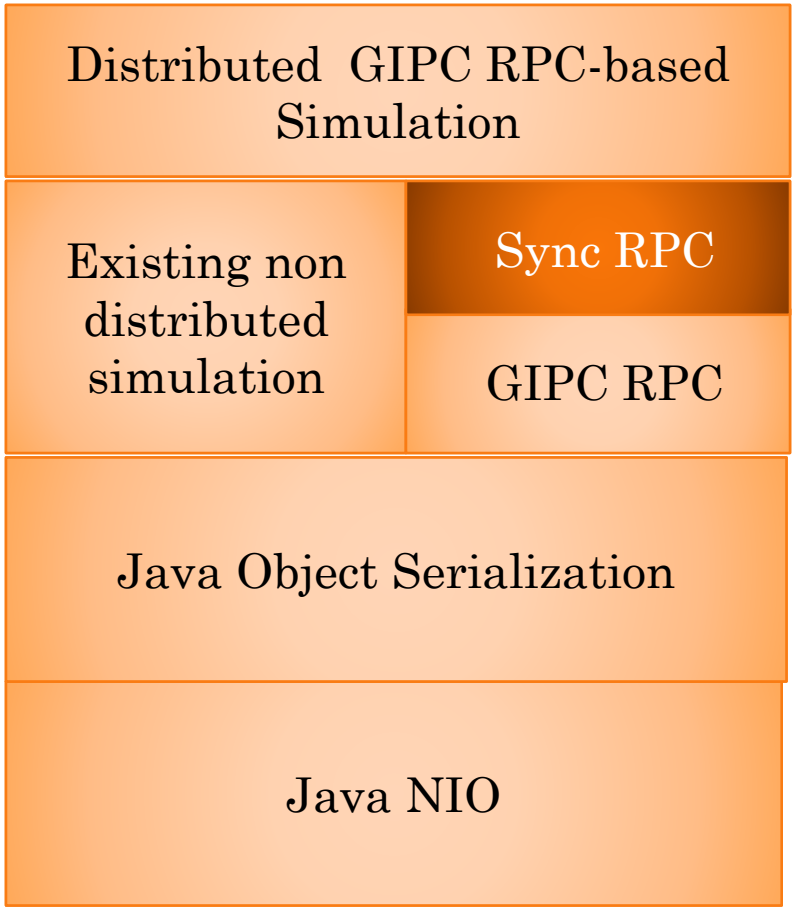
Assignment 3-Part 3: Distributed Cloud-based Experiments



Assignment 3-Part 1: Use Asynchronous RMI



Assignment 4: Custom RPC Component



Assignment 3-Part 2: Centralized, Non-Fault Tolerant Consensus Algorithm

Adaptive communication with consensus on current communication mechanism

Distributed Non-Blocking Simulation	Distributed RMI-based Simulation	Distributed GIPC-based Simulation

Assignment 5-Part 1: Understand and Trace Consensus Algorithms under Different Configurations/Algorithms

Change configuration, introduce faults and understand algorithms behavior

Example Replication

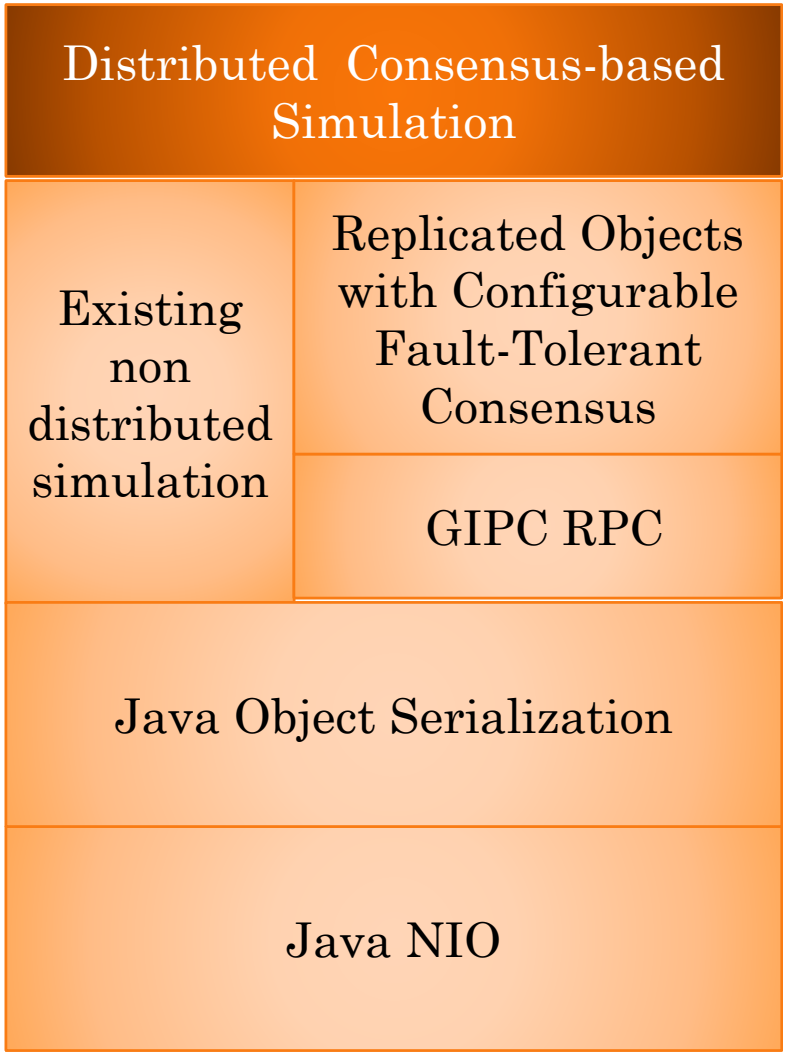
Replicated Objects with Configurable Fault-Tolerant Consensus

GIPC RPC

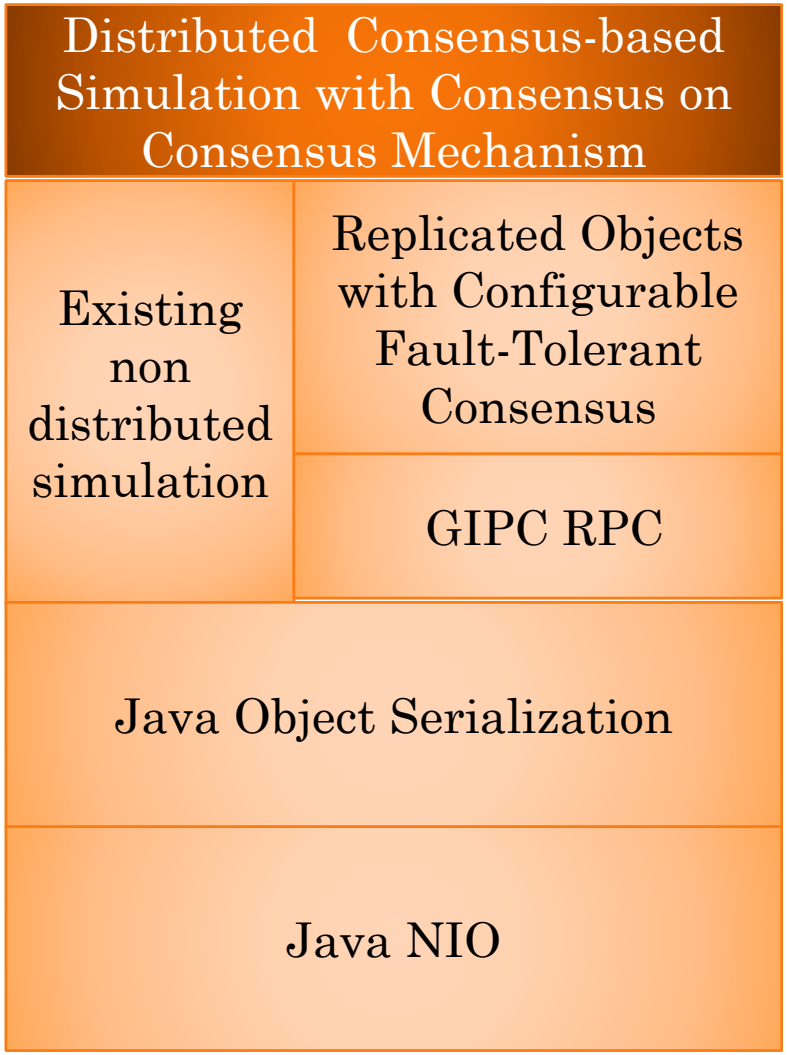
Java Object Serialization

Java NIO

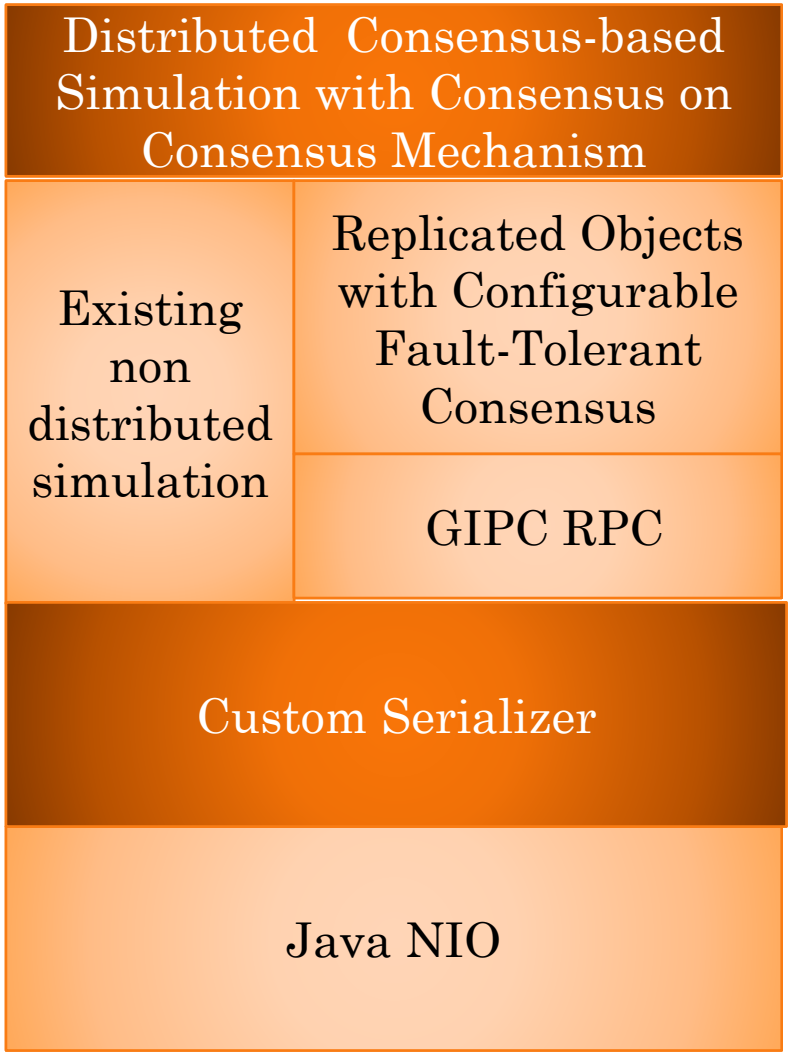
Assignment 5-Part 2: Use Configurable Consensus Mechanism for Sharing commands



Assignment 5-Part 3: Reach Consensus-Configuration Consensus



Assignment 6: Custom Serializer (Extra Credit)



Adding the Distributed Adjective

We will do an implementation-based coverage of selected concepts in design, use, and implementation of domain-independent distributed abstractions and algorithms (no text book)

Distributed operating systems covered in graduate courses and text books

Levels of abstraction and implementation?

Covered in Graduate course and several text books cover

Abstraction design

Abstraction use

Abstraction Implementation

Implementation of Algorithms and Associated Data Structures

Proof of Algorithms and Associated Data Structures

Earlier Version: Fall 13, 15

http://www.cs.unc.edu/~dewan/734/f11/index.html

Gmail - Inbox (33752) - pra... | unc.edu

Google | Search | More » | prasun...

Comp 734: Distributed Systems

Course Overview

This course will provide an implementation-oriented study of distributed systems. Some of the topics covered will include inter-process communication, group communication, synchronization, remote procedure call, peer to peer and centralized sessions, fire-walls, causal broadcast, atomic broadcast, scalability, fault tolerance, replication, and transactions/concurrency control. These are foundational concepts, which are becoming particularly relevant with the emerging areas of cloud computing and distributed games. These concepts will be introduced as layers in a general distributed infrastructure. Your projects will implement new layers and provide alternative implementations of some of the existing layers. When implementing a layer, you will act both as an application programmer, using abstractions of the layers below, and a systems programmer, defining abstractions for the layers above. The number of lines of code required by each layer will be relatively small; however the compositions of these layers will be complex.

First-year grads and several undergrads took this course

Earlier Version: Spring 17

www.cs.unc.edu/~dewan/533/s17/index.html

Comp 533: Distributed Systems

Course Overview

This course will provide a practical overview distributed systems. It will be driven by a series of implementation-based projects.

We will cover the design, implementation, performance, and applications of abstractions for sharing information among distributed processes. These sit between the network layer and application and include: distributed shared memory, byte and object communication, remote procedure call, and broadcast/multicast sessions. Some of the general issues we will address are naming; synchronization; extensibilty; and routing and consistency of broadcast/multicast messages.

We will overview but not focus on any of the specific applications of distributed abstractions such as the Web, distributed data mining, distributed data bases, file systems and distributed collaborative systems, which are covered in other courses. Our focus will be on foundational concepts applicable to all of these application areas. These concepts will be introduced as layers that sit above the OS and networks.

The assignments and lectures will have two related goals. The first is to expose you to standard implementation of the covered abstractions in Java environment and other distributed systems and discuss how these abstractions can be used in a realistic programs. The assignments addressing this goal will involve the use of only Java abstractions. The second is to expose you to the implementation of these abstractions. I have developed a Java-based teaching/research testbed, called GIPC (for Generalized Interprocess communication), that provides implementation of these abstractions. In the assignments addressing this goal, you will replace parts of this extendible system to understand implementation concepts. The result of the replacement exercised will be a system that is more sophisticated in many ways than the state of the art, and you are likely to use it for programming future distributed programs.

At the end of the course, you will have a basic understanding of how distributed software works, the potential uses of this software, the design and implementation space of distributed abstractions; the performance of alternative distributed abstractions and how to run experiments to measure performance. As a distributed program is also a parallel program, you will also sharpen your understanding of threads and thread synchronization. Because of the emphasis on assignments, you will gain practice with the use and implementation of

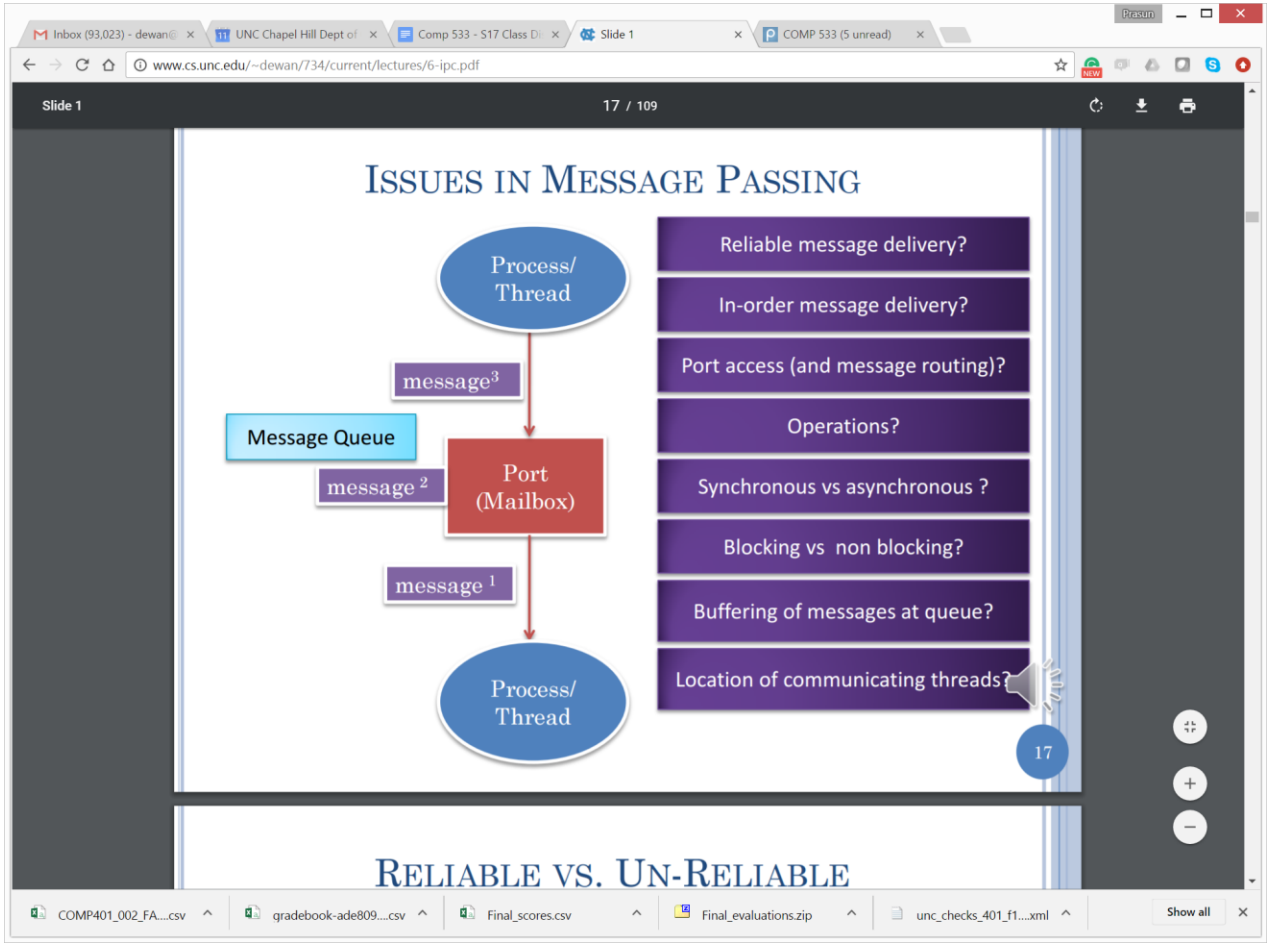
First assignment was a problem, assignment 3 needs to be cleaned up, auto grading needed, relatively little and simple code needs to be written for regular assignments, compared to Beau's project at least (distributed debugging is the challenge)

Learning Resources

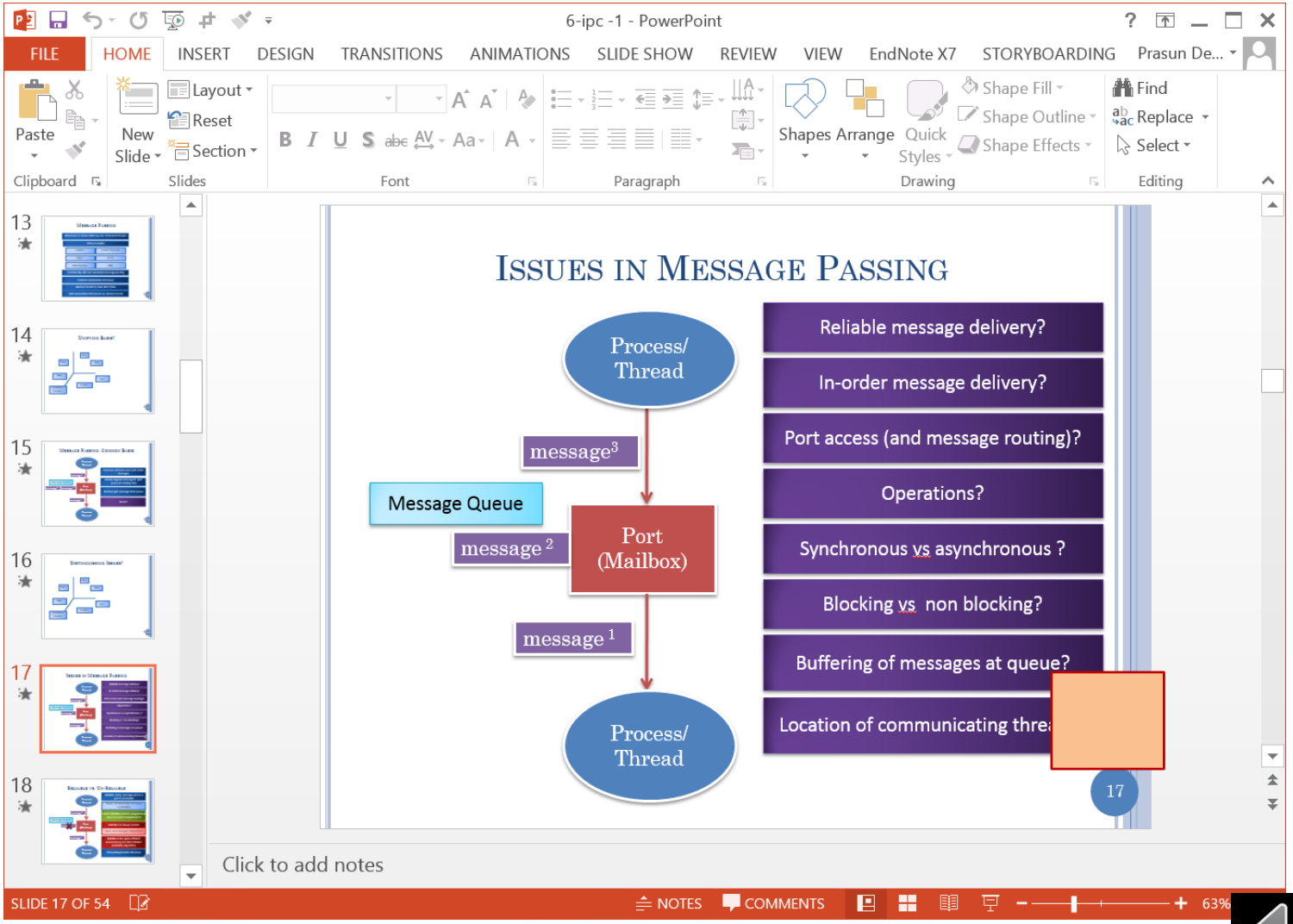
No textbook!

Alternatives?

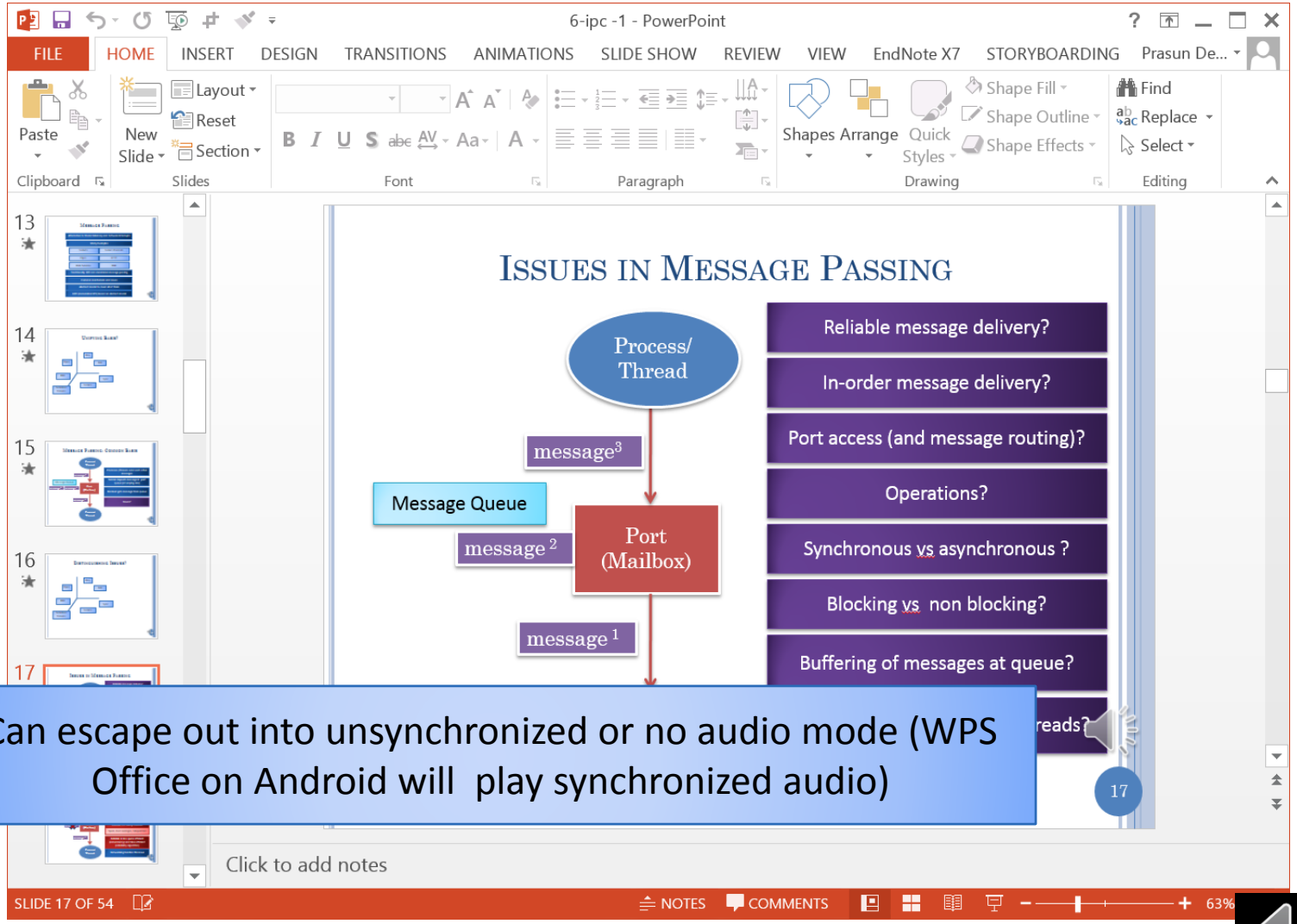
PDF of Slides



PowerPoint of Slides

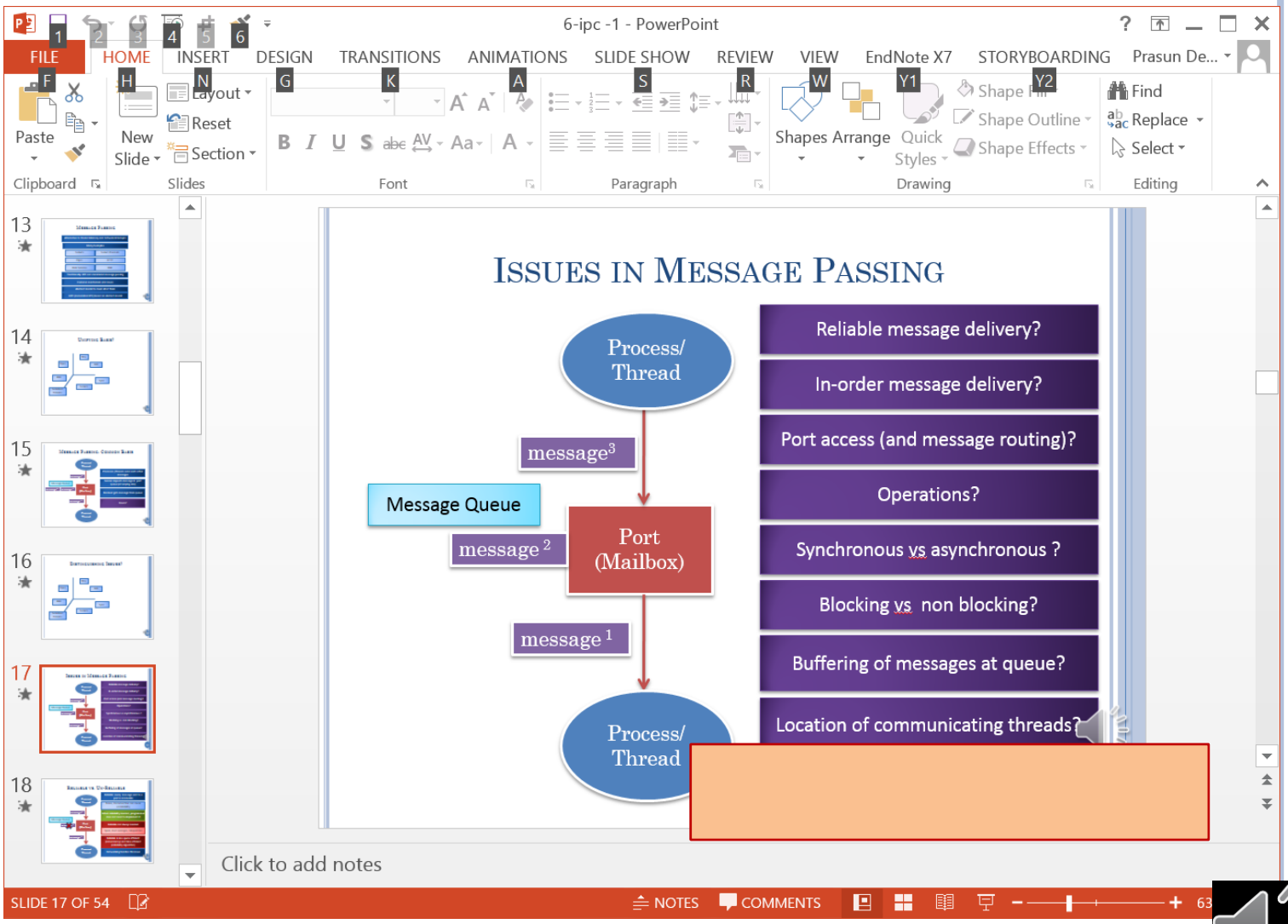


Slide Show → Synchronized Recording and Animations

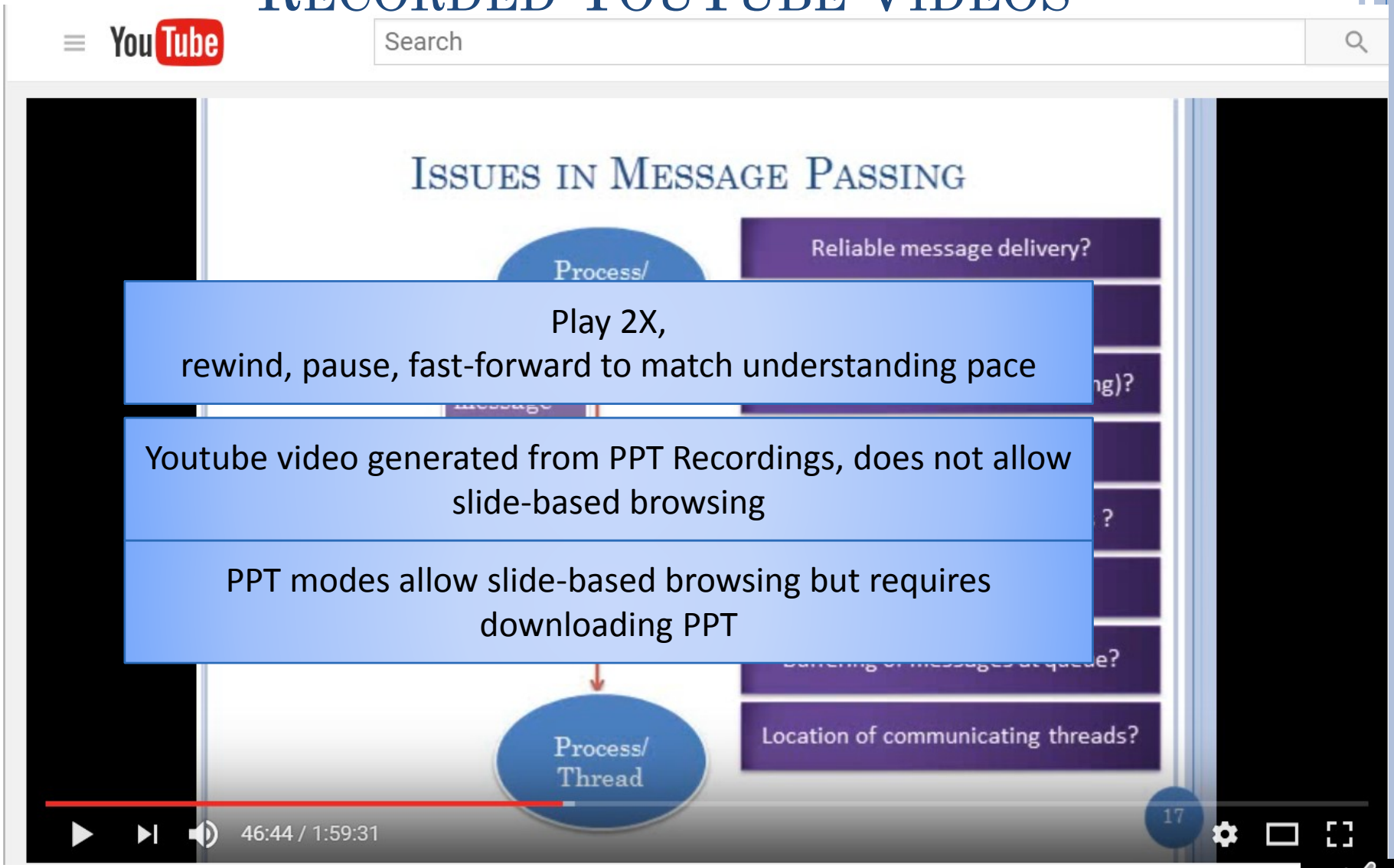


Can escape out into unsynchronized or no audio mode (WPS Office on Android will play synchronized audio)

PowerPoint Slides With Unsynchronized Recordings and Media Control



Recorded YouTube Videos



Play 2X,
rewind, pause, fast-forward to match understanding pace

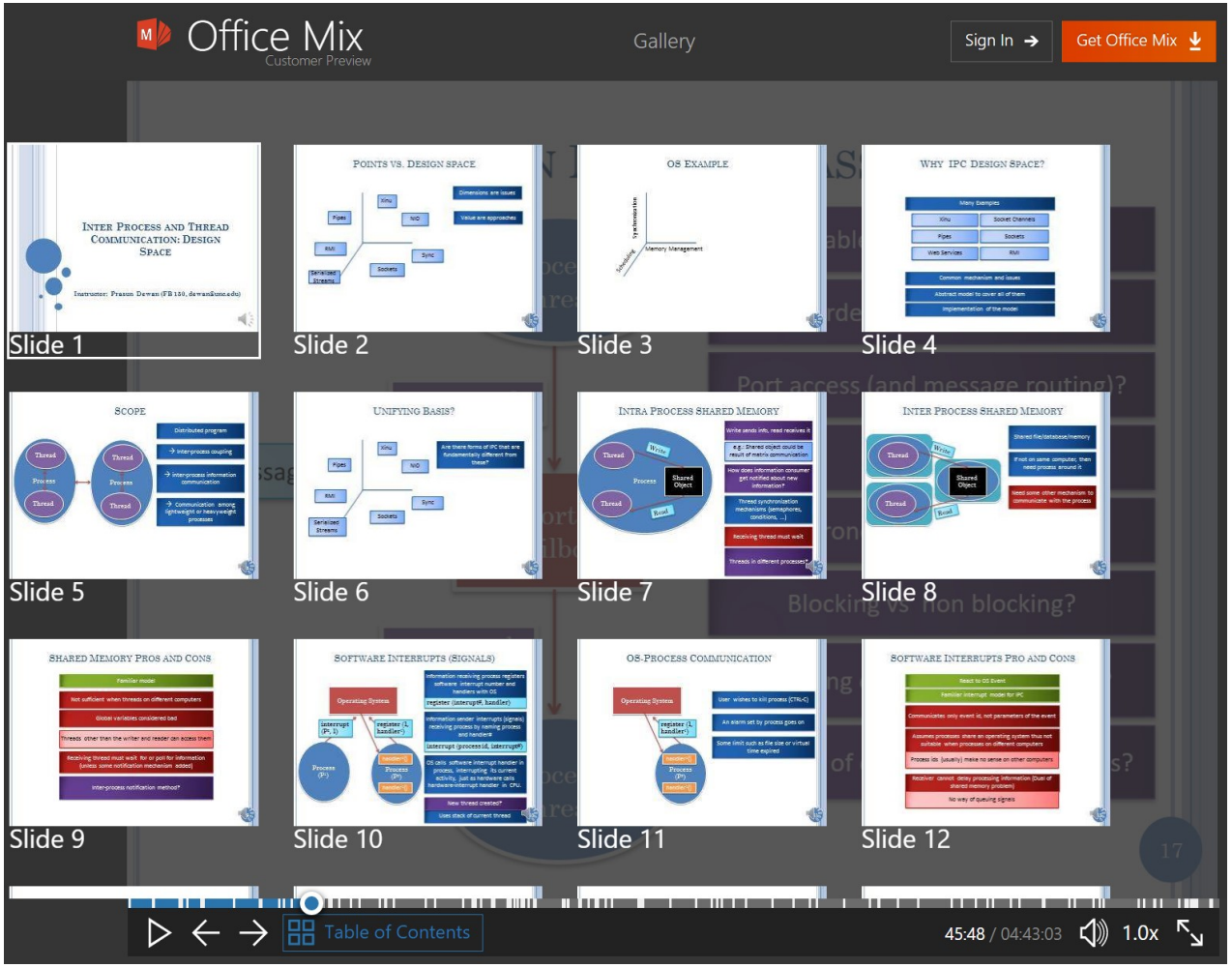
Youtube video generated from PPT Recordings, does not allow slide-based browsing

PPT modes allow slide-based browsing but requires downloading PPT

Office Mix

Slide markers

Slide-Based Browsing



Video Nature

Long pauses, you may know the answer

Cannot hear student answer

Audio is not the fastest way to get information, specially when studying for an exam

Recordings of live lectures with q/a rather than 15 minute lessons

Can fast forward

You can get a clue from my answer

Written Documents

- Interprocess Communication
  - Shared Memory
  - Software Interrupt
  - Message Passing
    - Reliability of Messages
    - Order
    - Access
    - Remote Assignment vs Procedure Call
    - Synchronous vs Asynchronous
    - Buffering of Messages
    - Pipes
    - Selectivity of Receipt
    - Integration with Programming Language
    - Integration with I/O
  - Xinu Low-Level Message Passing
    - Semantics
    - Implementation
- Communication across a Network

Incomplete

Concise

DETAILED ASSIGMENTS

Consult this section only if you are stuck trying to implement some aspect of the assignment. You will, of course, learn more by figuring out the implementation structure on your own.

You essentially have to extend the "coupled simulation" program using the distributed pattern illustrated in the "meaning of life" distributed program.

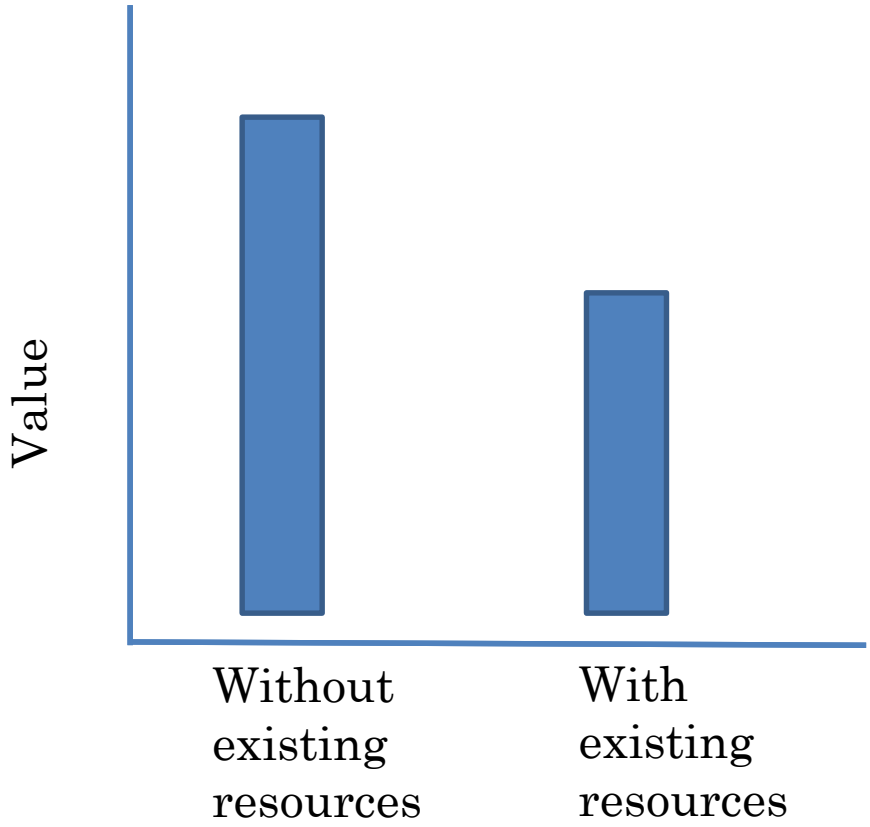
The trace of the coupled simulation program illustrates how it couples the two simulation models.

The following steps occur before any interaction occurs:

```
I***{main}(AddedPropertyChangeListener) EvtSrc(AHalloweenCommandProcessor)

I***{main}(AddedPropertyChangeListener) EvtSrc(AHalloweenCommandProcessor)

I***{main}(AddedPropertyChangeListener) EvtSrc(AHalloweenCommandProcessor)
coupledsims.ASimulationCoupler@3445341

I***{main}(AddedPropertyChangeListener) EvtSrc(AHalloweenCommandProcessor)
coupledsims.ASimulationCoupler@14cc1f5c
```

The first two steps are ObjectEditor registering view observers (whose toString() methods return the empty strings) with each of the two coupled models described by the class,

What do we do in class?



Live Lectures only on new material

Scheduling time to view recording an issue

Can listen to recordings in class using headphones

What do we do in class?

Homework

Deep thinking done solo?

Limited discussion with classmates?

Can do debugging

What do we do in class?

Quizzes: Test that you understood support material

One hour testing, puts pressure

May not have discipline to access material

No time pressure, open book, collaborative, submittable later

What do we do in class?

Participate in lectures/discussions

Do quizzes (collaboratively)

Troubleshoot or implement assignments

Listen to recorded lectures

Choice is good

Choice can paralyze or be excuse to procrastinate

What do we do in class? (Review)

Participate in lectures/discussions

Do quizzes (collaboratively)

Troubleshoot or implement assignments

Listen to recorded lectures

Choice is good

Choice can paralyze or be excuse to procrastinate

Progress Diary

Participate in lectures/discussions	Explain some concept that was presented or discussed
Do quizzes (collaboratively)	Explain some concept on which you took a quiz
Troubleshoot or implement assignments	Explain some progress you made on an assignment or some issue that prevented progress
Listen to recorded lectures	Explain some concept that was presented or discussed

Describe at least one specific piece of work you did in each class

Q/A Diary

Give all (correct/incorrect/partially correct) answers to questions asked together with the questions

Diary Posting Mechanism

Piazza post visible to instructors (and possibly students also)

Diary Example and Format

Date: 1/11

Activity: Learned (from live lecture/recorded lecture/quiz) that a selector can wait for a specific set of NIO operations to complete.

QA:
Question: Why allow multiple pending operations?
My answer: To increase concurrency without requiring thread overhead.

Format is important as I plan to do some automatic processing of diaries

Can have multiple activities and answers on a specific date

Why Class Participation and diaries

Instructor needs feedback
Learn to speak in public (I will ask questions every one can answer such as what was interesting in the material you studied(
Oral “exam” over whole semester
Retain material better, logical vs physical attendance
Recruiters regularly ask about oral and verbal skills
NSF Survey asked what was being done to improve these skills

More Oral Skills

< 5 minute Videos of Demos
Safeguard against lack of and errors in auto-grading

Grade Distribution

Midterm	25%
Final	15%
Regular Collaborative Open-Book Quizzes	10%
Answers to In-Class Q/A (based on diary)	5%
Class activity report (based on diary)	5%
5 Individual Assignments	40%
Extra Credit Serialization Individual Assignment	12%
Extra Credit Serialization Open-Book Collaborative Quizzes	3%

Finish assignments (> 90%) and median on other parts assures some form of B or better

Comparing the Two Approaches

Traditional: Live Lectures + Exams

Mixed Mode: Live Lectures + Exams + Quizzes + Assignments + Diaries.

Live Lectures + Exam

Much lower instructor effort

Illusion of or actual understanding

1/2/3 Exams show gaps in retention and understanding

Active Learning

Continuous demonstration of gaps in retention and understanding

Continuous fixes to these problems

At the end of course may feel less comfortable with the experience and unsure about your understanding

Frustration vs Creativity

Tim Harford *at* TEDGlobal>London

How frustration can make us more creative

Details Transcript Footnotes Comments

Challenges and problems can derail your creative process ... or they can make you more creative than ever. In the surprising story behind the best-selling solo piano album of all time, Tim Harford may just convince you of the advantages of having to work with a little mess.

This talk was presented at an official TED conference, and was featured by our editors on the home page.

Recorded

September 2015 at TEDGlobal>London

Related tags

Cognitive
Science
Collaboration
Composing

How frustration can make us more creative

A pianist working with a defective piano created the best-selling solo album, after initially refusing to use it.

Studying from a harder to read font resulted in better results.

Diverse groups created better solutions but were more frustrated and less comfortable with their results.

Lecturing vs. Learning Experience

Midterm	25%
Final	15%
Regular Collaborative Open-Book Quizzes	10%
Answers to In-Class Q/A (based on diary)	5%
Class activity report (based on diary)	5%
5 Individual Assignments	40%
Extra Credit Serialization Individual Assignment	12%
Extra Credit Serialization Open-Book Collaborative Quizzes	3%

Assignment Help

Can discuss solutions with each other at a high level

Not at the code level

Sharing of code is honor code violation

Can help each other with debugging as long as it does not lead to code sharing

Assignments may contain solution in English (read only if stuck)