Last Updated: 2021-09-08 Wed 13:54

CS 5451 Assignment 1: Basic Parallel Architecture and Communication

- **Due: Fri 9/18/2021 by 11:59 pm**
- *Approximately 10% of total grade*
- Submit to **Gradescope** in PDF format
- You may work in groups of 2 and submit one assignment per group.

CHANGELOG: Empty

Table of Contents

Note: some of the math fonts used throughout the assignment may not display correctly. I have tested the display on Google Chrome and seen it work when connected to the network. Notify Prof Kauffman via email if it does not render properly.

1 (20pts) Problem 1: Store-and-forward Routing (Grama 2.26)

Consider the routing of messages in a parallel computer that uses store-and-forward routing. In such a network, the cost of sending a single message of size m from P_{source} to $P_{destination}$ via a path of length d is $t_s + t_w \times d \times m$. An alternate way of sending a message of size m is as follows. The user breaks the message into k parts each of size m/k, and then sends these k distinct messages one by one from P_{source} to $P_{destination}$. For this new method, derive the expression for time to transfer a message of size m to a node d hops away under the following two cases:

A. Assume that another message can be sent from P_{source} as soon as the previous message has reached the next node in the path.
B. Assume that another message can be sent from P_{source} only after the previous message has reached $P_{destination}$.

For each case, comment on the value of this expression as the value of k varies between 1 and m. Also, what is the optimal value of k if t_s is very large, or if $t_s = 0$?

Clarification: To start sending any message, a cost of t_s must be paid. For example sending $m = 10$ words in its entirety costs

$$t_s + t_w \times d \times 10$$

Sending the same message broken into two 5-word chunks costs

$$(t_s + t_w \times d \times 5) + (t_s + t_w \times d \times 5)$$

if the sending does not overlap.

2 (10pts) Problem 2: Shared vs. Distributed Memory Architectures (Grama 2.7, 2.8)

Outline the major differences between a shared memory parallel computer and a distributed memory parallel computer. Discuss the following in your answer.

A. Overview the differences in the basic architecture of these two types of machines with respect to memory access.
B. Describe how the programming models are different between these two paradigms due to the differences in memory access.
C. Report the relative abundance of one architecture versus the other and where they typically appear "in the wild".
D. Which architectures scales better as more processors are incorporated. For example, discuss the feasibility of building a 1024 processor computer with 1024 GB of RAM as a distributed memory machine versus as a shared memory machine.

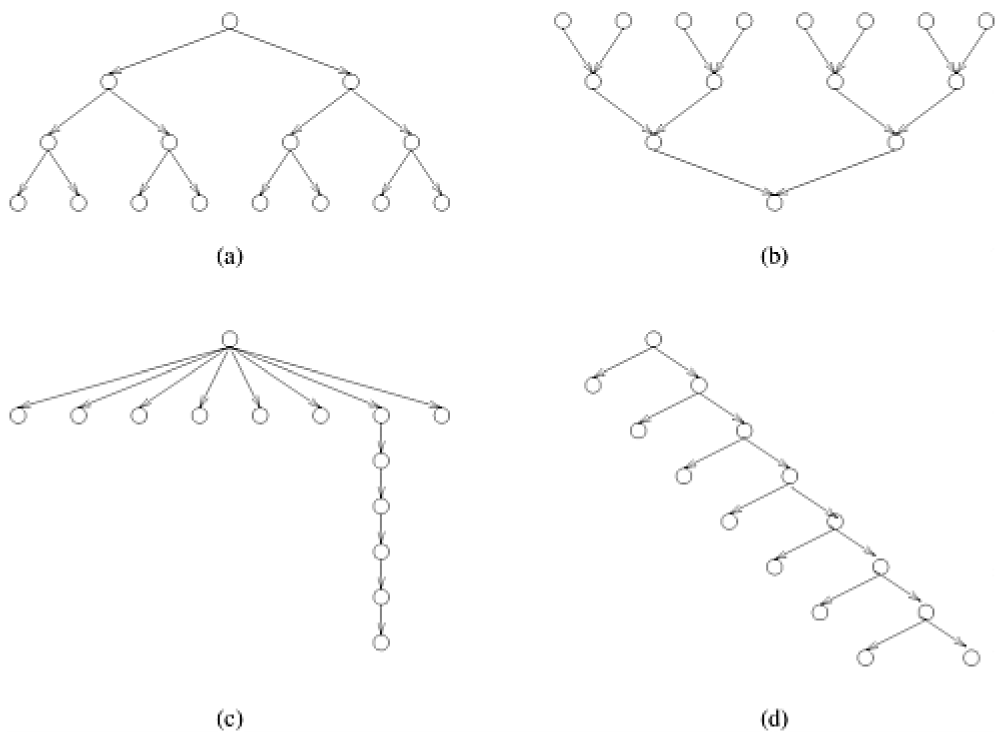
3 (10pts) Problem 3: Task-dependence Graphs (Grama 3.2)

For the task graphs given in Figure 3.42 (below), determine the following:

A. Maximum degree of concurrency.
B. Critical path length.
C. Maximum achievable speedup over one process assuming that an arbitrarily large number of processes is available.
D. The minimum number of processes needed to obtain the maximum possible speedup.
E. The maximum achievable speedup if the number of processes is limited to 2, 4, and 8.

Assume that each task node takes an equal amount of time to execute.

Figure 3.42. Task-dependency graphs for this Problem.



4 (20pts) Problem 4: Parallel Histogram

This problem is a variant of Grama 3.19 + 3.20 which discusses Bucket Sort but focuses on the simpler task of computing Histograms.

Consider parallelizing a simple histogram algorithm on a **distributed memory computer**.

- Input consists of an array `A[]` of `N` random integers in the range `{0...(R-1)}`.
- The algorithm should compute the array of counts `H[]` where element `H[i]` is the total number of times integer `i` appeared in input array `A[]`.
- All `P` processors on the parallel computer are assumed to have access to the entire array `A[]` which can be loaded from permanent storage. However if the size `N` gets large enough, only parts of `A[]` can be held in memory for an individual processor.

- At the end of the algorithm execution, at least one processor must contain the entire array `H[]` which may involve some communication.

EXAMPLE:

- Input: `R=4` and `A[]` of size `N=10` is `A[] = {0, 2, 1, 3, 2, 1, 1, 0, 0, 1}`
- Output: `B[]` of size `R=4` is `B[] = {3, 4, 2, 1}` as `B[i]` is the number of occurrences of `i` throughout `A[]`

Describe two ways to decompose this problem into a parallel program. Compare these two.

A. Describe a decomposition based on **partitioning the input data** (i.e., the array `A[]`) and an appropriate mapping onto p processes. Describe briefly how the resulting parallel algorithm would work.
B. Describe a decomposition based on **partitioning the output data** (i.e., the array `H[]`) and an appropriate mapping onto p processes. Describe briefly how the resulting parallel algorithm would work.
C. Discuss the advantages and disadvantages of these two parallel formulations. Describe circumstances under which each is preferred. Such circumstances should consider
    - `P` the number of processors
    - `R` the number of different data possibilities (size of `H[]`)
    - `A[]` the input array and `n` its size
    - Potentially also the cost to communicate between the processors

5 (20pts) Problem 5: Parallel One-to-All Broadcast in 2D Torus

Assume processor 0 has a message that must be sent to all other processors in a distributed memory parallel computer. Describe an efficient algorithm to do this in a 2D Torus (wrap-around mesh) with `[RC]` rows and `[RC]` columns.

Account for the following in your answers.

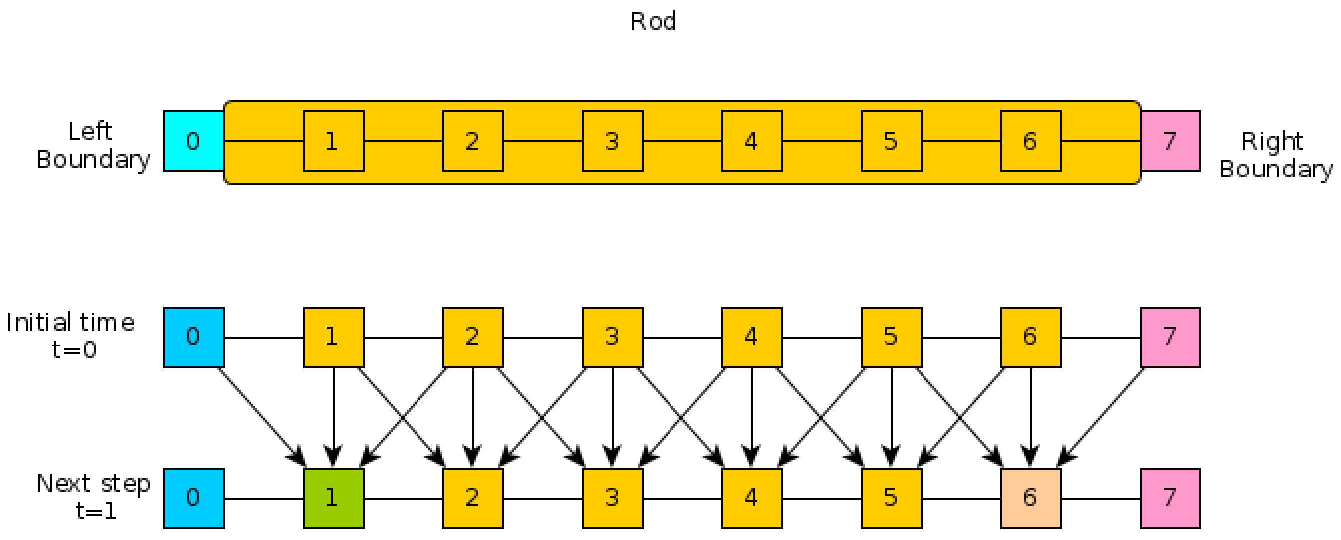
- A processor can send a message to any other processor. Any processor between the source and destination does not receive the message, only forwards it along the path. However, once a processor has actually received the message, it can subsequently be a source and send it to other processors.
- Processors can be indexed with 2D coordinates as in processor (0,2) sends a message to processor (5,4)
- Give rough pseudocode for how this algorithm will look
- Give a cost estimate of this communication in terms of the number or rows `[RC]` and columns `[RC]` in the network. The cost of a single communication between nodes that are d hops away is

$$t_{single} = t_s + t_w \times m \times d$$

  You may assume that `[RC]` and `[RC]` are powers of two in your analysis to simplify the expression for the total communication time t_{comm}.

6 (20pts) Problem 6: Bring the Heat

Consider a simple physical simulation of heat transfer. A rod of material is insulated except at its ends to which are attached two constant sources of heat/cold.



The rod is broken into discrete chunks, sometimes referred to as finite elements, and it is assumed that throughout each element the temperature is uniform. The reservoir elements of heat/cold at the left and right remain at a constant temperature. For the other elements, the temperature is updated at each time step by examining the temperature of its neighbors compared to itself. This dependence is indicated in the lower part of the diagram and the exact update for an element is given in the code below.

The following C code implements a simple version of this simulation. A row of the matrix `H` contains the temperature of all elements in the rod at a time step with the leftmost and rightmost elements remaining constant. Ultimately, the program computes the entire matrix `H` which contains temperatures of all elements at all time steps. Examine this code carefully.

```
#include <stdio.h>
#include <stdlib.h>
//  HEAT TRANSFER SIMULATION
//
//  Simple physical simulation of a rod connected at the left and right
//  ends to constant temperature heat/cold sources.  All positions on
//  the rod are set to an initial temperature.  Each time step, that
//  temperature is altered by computing the difference between a cells
//  temperature and its left and right neighbors.  A constant k
//  (thermal conductivity) adjusts these differences before altering
//  the heat at a cell.  Use the following model to compute the heat
//  for a position on the rod according to the finite difference
//  method.
//
//     left_diff  = H[t][p] - H[t][p-1];
//     right_diff = H[t][p] - H[t][p+1];
//     delta = -k*( left_diff + right_diff )
//     H[t+1][p] = H[t][p] + delta
//
//  Substituting the above, one can get the following
//
//    H[t+1][p] = H[t][p] + k*H[t][p-1] - 2*k*H[t][p] + k*H[t][p+1]
//
//  The matrix H is computed for all time steps and all positions on
//  the rod and displayed after running the simulation.  The simulation
//  is run for a fixed number of time steps rather than until
//  temperatures reach steady state.

int main(int argc, char **argv){
  int max_time = 50;            // Number of time steps to simulate
  int width = 20;               // Number of cells in the rod
  double initial_temp = 50.0;   // Initial temp of internal cells
  double L_bound_temp = 20.0;   // Constant temp at Left end of rod
  double R_bound_temp = 10.0;   // Constant temp at Right end of rod
  double k = 0.5;               // thermal conductivity constant
  double **H;                   // 2D array of temps at times/locations

  // Allocate memory
  H = malloc(sizeof(double*)*max_time);
  int t,p;
  for(t=0;t<max_time;t++){
    H[t] = malloc(sizeof(double*)*width);
  }

  // Initialize constant left/right boundary temperatures
  for(t=0; t<max_time; t++){
    H[t][0] = L_bound_temp;
    H[t][width-1] = R_bound_temp;
  }

  // Initialize temperatures at time 0
  t = 0;
  for(p=1; p<width-1; p++){
    H[t][p] = initial_temp;
  }

  // Simulate the temperature changes for internal cells
  for(t=0; t<max_time-1; t++){
    for(p=1; p<width-1; p++){
      double left_diff  = H[t][p] - H[t][p-1];
      double right_diff = H[t][p] - H[t][p+1];
      double delta = -k*( left_diff + right_diff );
      H[t+1][p] = H[t][p] + delta;
    }
  }

  // Print results
  printf("Temperature results for 1D rod\n");
  printf("Time step increases going down rows\n");
  printf("Position on rod changes going accross columns\n");

  // Column headers
  printf("%3s| ","");
  for(p=0; p<width; p++){
    printf("%5d ",p);
  }
  printf("\n");
  printf("%3s+-","---");
  for(p=0; p<width; p++){
    printf("------");
  }
  printf("\n");
  // Row headers and data
  for(t=0; t<max_time; t++){
    printf("%3d| ",t);
    for(p=0; p<width; p++){
      printf("%5.1f ",H[t][p]);
    }
    printf("\n");
  }

  return 0;
}
```

Answer the following questions about how to parallelize this code.

A. A typical approach to parallelizing programs is to select a loop and split iterations of work between available processors. Describe how one might do this for the heat program. Make sure to indicate any loops for which this approach is not feasible and any which seem more viable to you.
B. Describe how you would divide the data for the heat transfer problem among many processors in a distributed memory implementation to facilitate efficient communication and processing. Describe a network architecture that seems to fit this problem well and balances the cost of the network well.

7 Submission Instructions

- Complete the problems in each section labelled **Problem**
- This is a **pair assignment:** you may select one partner to work with on this assignment as a group of 2. You may also opt to work alone as a group of 1.
- **Only 1 member of your group should submit the HW Writeup on Gradescope.** If you are working in a partnership, **add your partner after submitting** as described below.

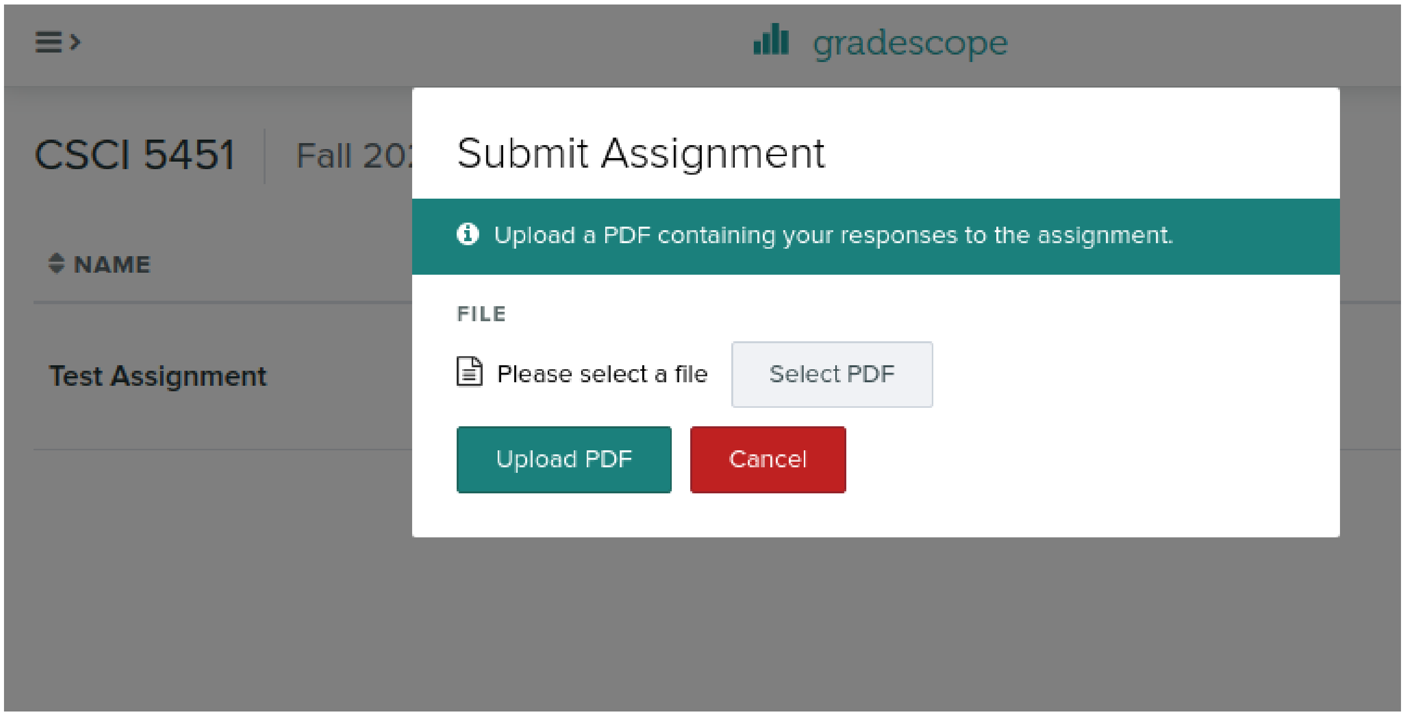
- Make sure the HW Writeup has all group member's information in it:

```
CS 5451 HW 1
Group of 2
Turanga Leela tleela4@umn.edu
Philip J Fry pfry99@umn.edu
```

- **Submit a PDF (portable document format, .pdf)** of your work. Whatever program you write your answers in (Microsoft Word, Apple Words, Google Docs, Latex, etc.) make sure you can export or "save as" a PDF.
- **Submit** your work to Gradescope under Assignment 1
  - Log into Gradescope
  - Assignment 1
  - Select your PDF for upload
  - Use the Gradescope system to indicate which of your answers are on which page of your PDF submission - this will make things easier on your grader (you want a happy grader).
  - After uploading, **add your partner as a group member** if you had one.

7.1 Picture Sequence of Uploading an Assignment

Selecting a PDF to upload

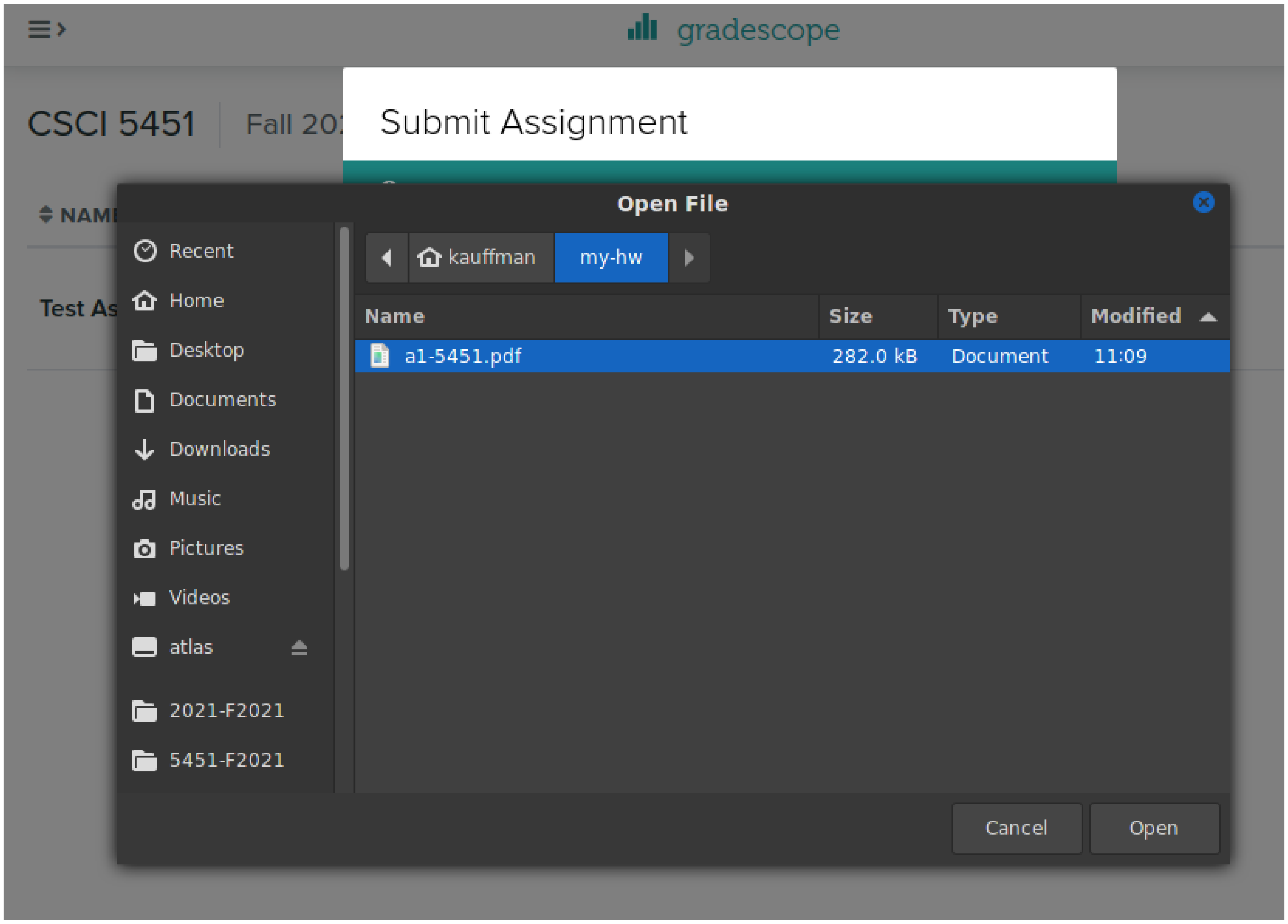


Submit Assignment

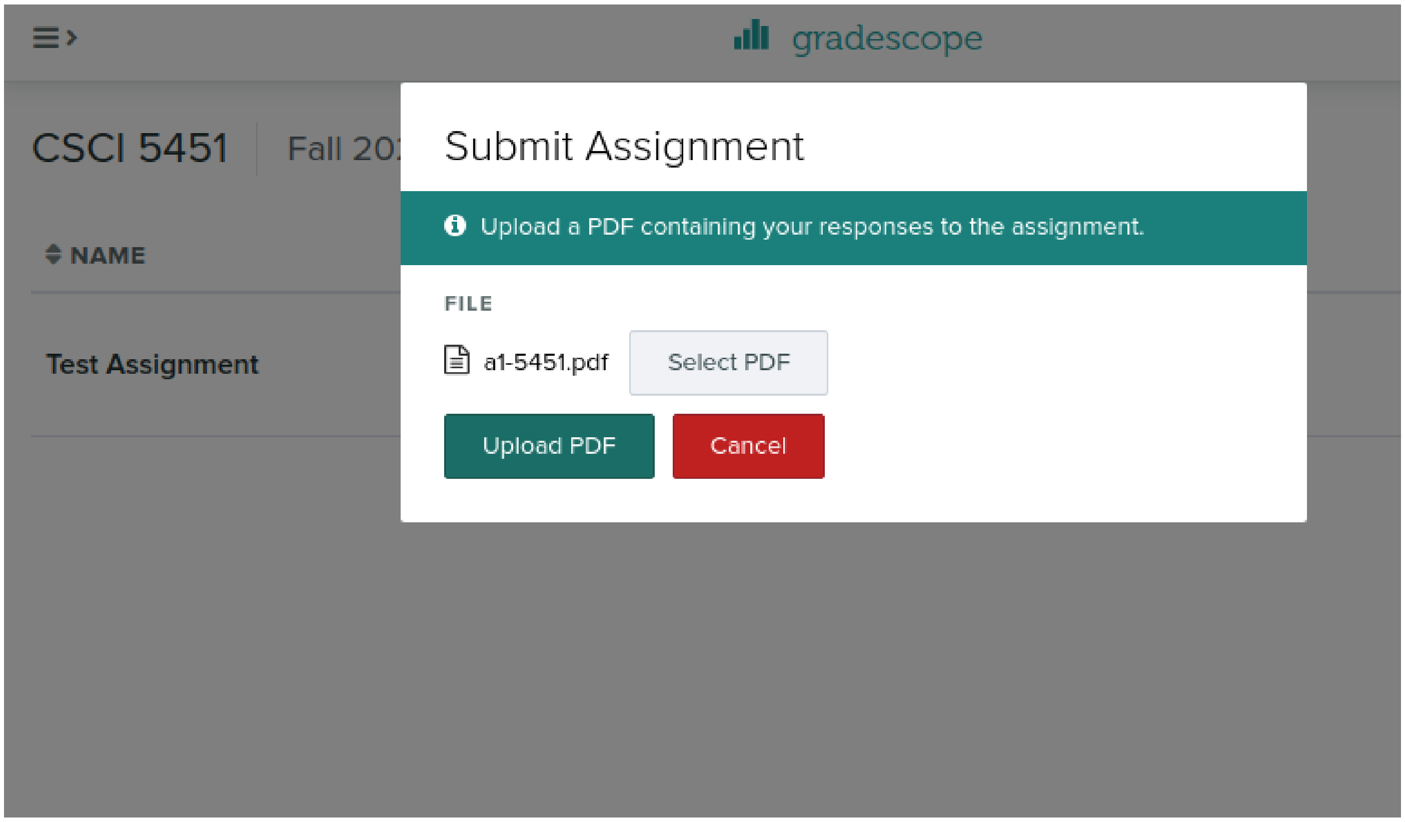
Upload a PDF containing your responses to the assignment.

FILE

Please select a file



Name	Size	Type	Modified
a1-5451.pdf	282.0 kB	Document	11:09



gradescope

CSCI 5451 Fall 20

NAME

Test Assignment

Submit Assignment

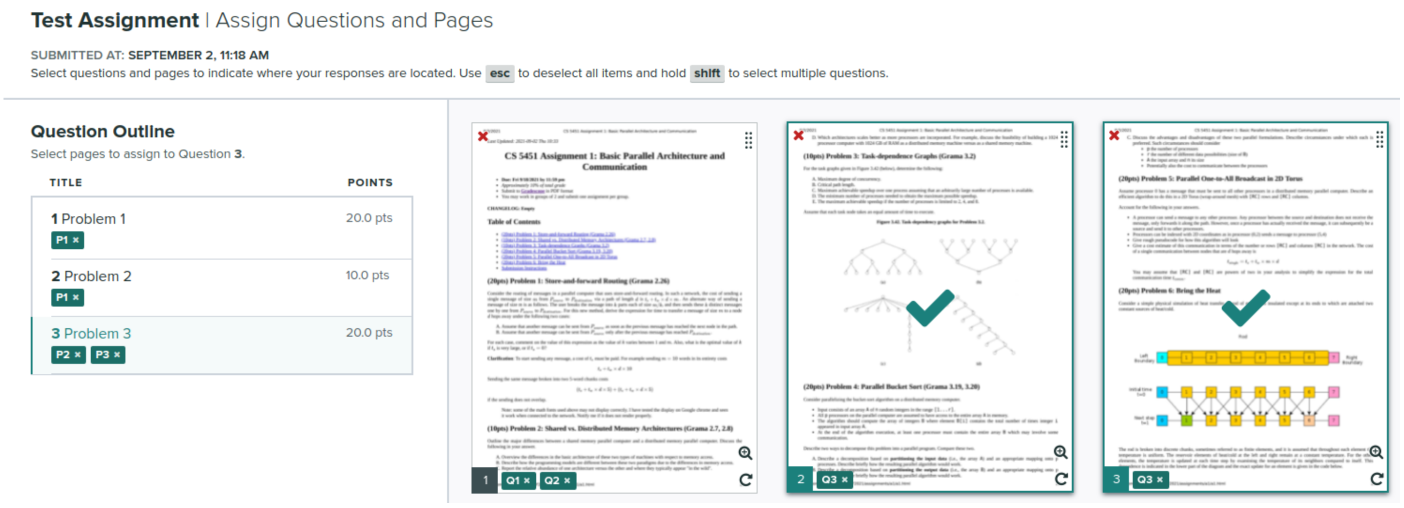
Upload a PDF containing your responses to the assignment.

FILE

a1-5451.pdf Select PDF

Upload PDF Cancel

Indicate problem answers on each page



Test Assignment | Assign Questions and Pages

SUBMITTED AT: SEPTEMBER 2, 11:18 AM

Select questions and pages to indicate where your responses are located. Use esc to deselect all items and hold shift to select multiple questions.

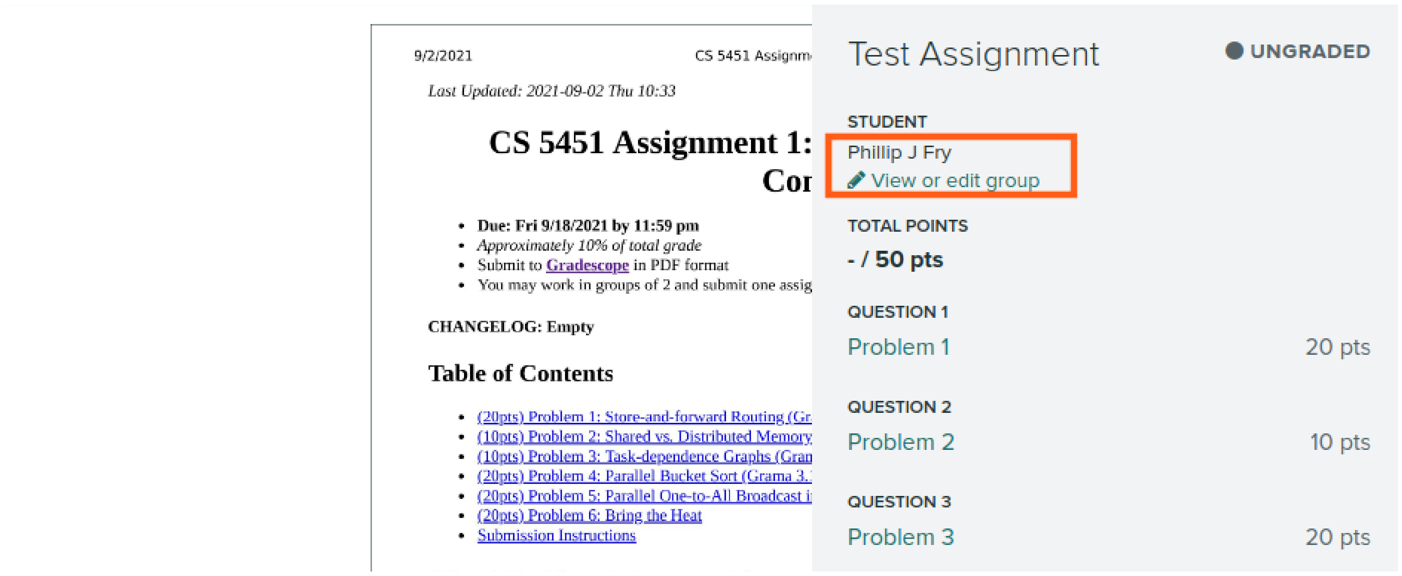
Question Outline

Select pages to assign to Question 3.

TITLE	POINTS
1 Problem 1 (P1)	20.0 pts
2 Problem 2 (P1)	10.0 pts
3 Problem 3 (P2, P3)	20.0 pts

7.2 Adding a Group Member on Gradescope

Submitter should edit the group associated with the submission



9/2/2021 CS 5451 Assignm

Last Updated: 2021-09-02 Thu 10:33

CS 5451 Assignment 1:
Co

- **Due: Fri 9/18/2021 by 11:59 pm**
- *Approximately 10% of total grade*
- Submit to Gradescope in PDF format
- You may work in groups of 2 and submit one assig

CHANGELOG: Empty

Table of Contents

- (20pts) Problem 1: Store-and-forward Routing (Gr
- (10pts) Problem 2: Shared vs. Distributed Memory
- (10pts) Problem 3: Task-dependence Graphs (Gran
- (20pts) Problem 4: Parallel Bucket Sort (Grama 3.
- (20pts) Problem 5: Parallel One-to-All Broadcast i
- (20pts) Problem 6: Bring the Heat
- Submission Instructions

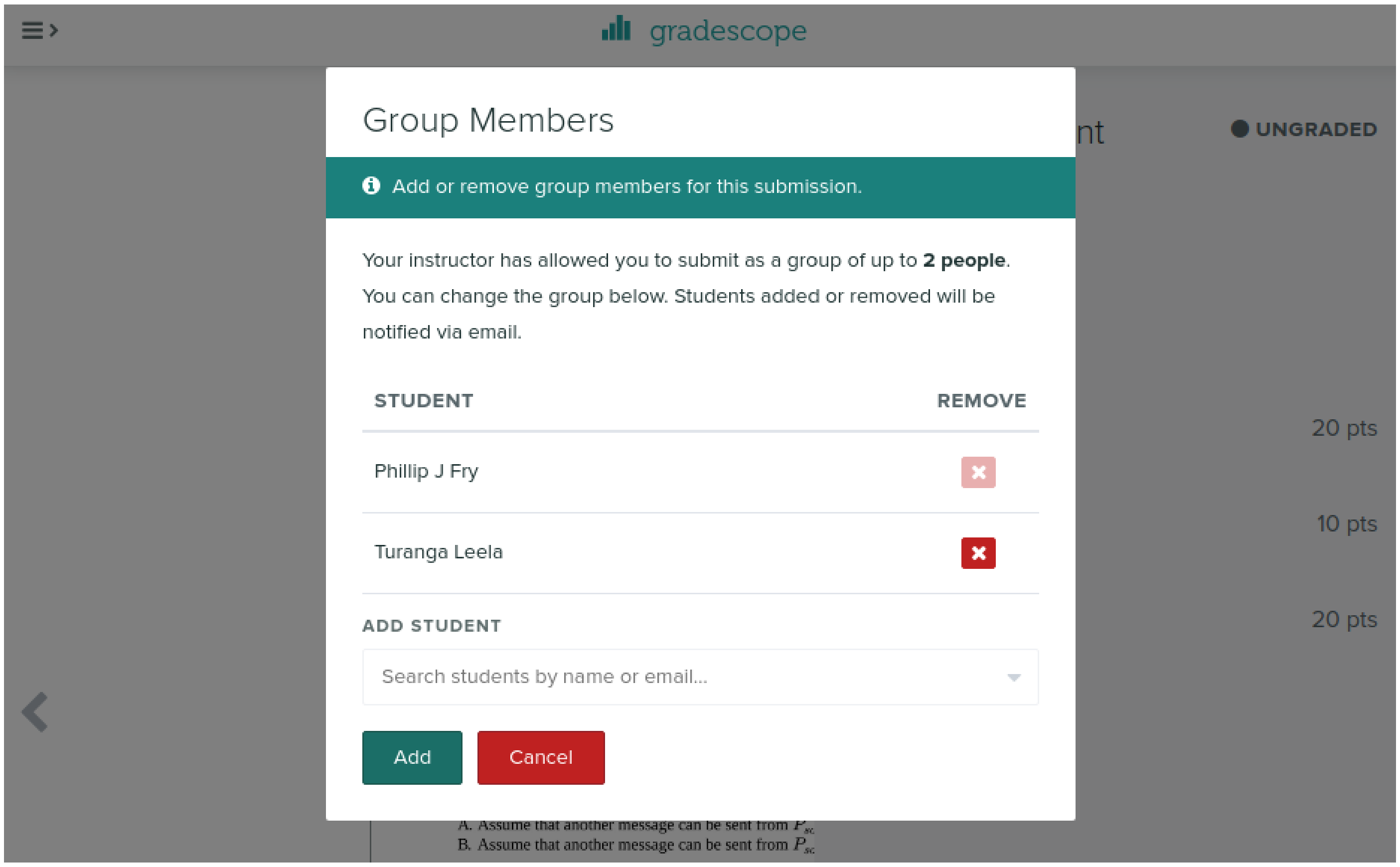
Test Assignment ● UNGRADED

STUDENT
Phillip J Fry
View or edit group

TOTAL POINTS
- / 50 pts

QUESTION 1 Problem 1	20 pts
QUESTION 2 Problem 2	10 pts
QUESTION 3 Problem 3	20 pts

Find your partner in the search box



gradescope

Group Members

Add or remove group members for this submission.

Your instructor has allowed you to submit as a group of up to **2 people**. You can change the group below. Students added or removed will be notified via email.

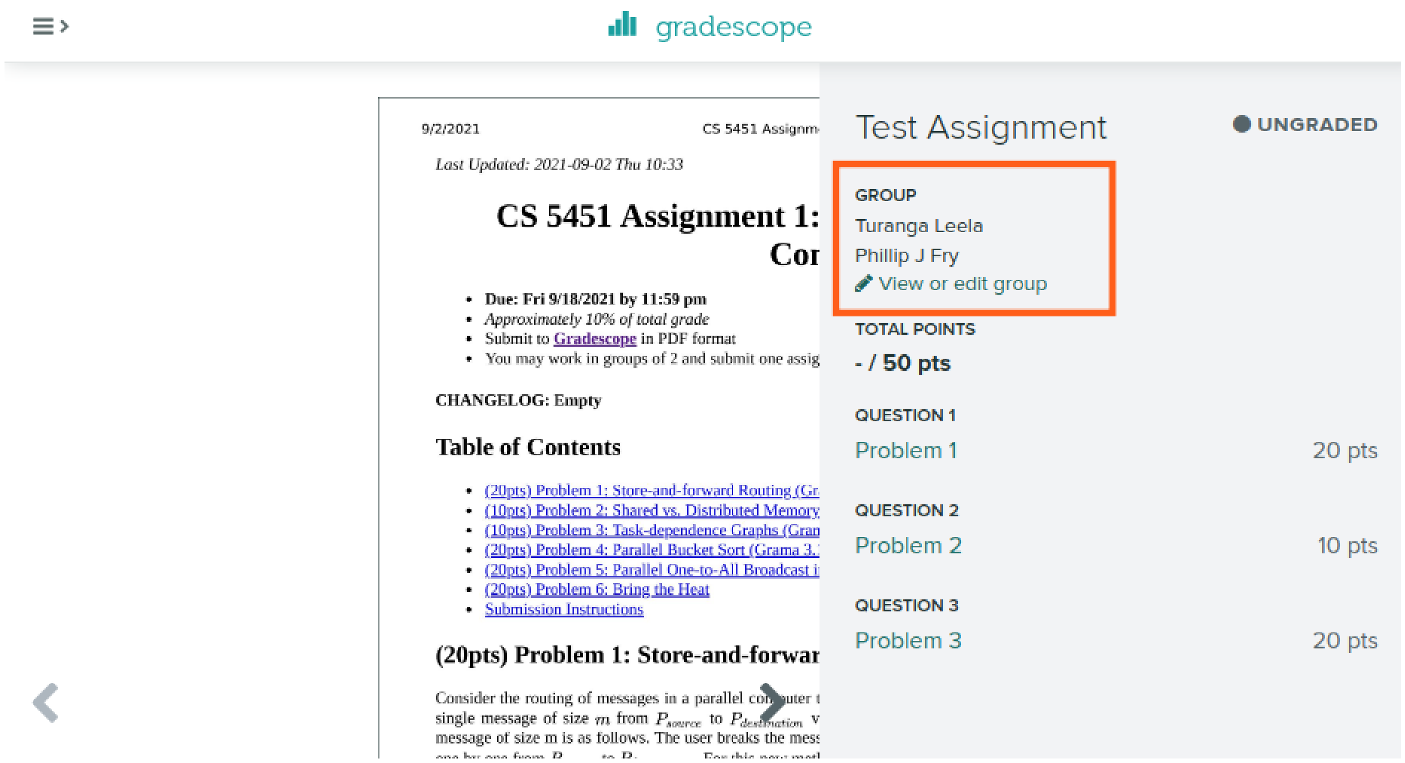
STUDENT	REMOVE
Phillip J Fry	✖
Turanga Leela	✖

ADD STUDENT

Search students by name or email...

A. Assume that another message can be sent from P_{sc}
B. Assume that another message can be sent from P_{sc}

Ensure your partner is listed



gradescope

9/2/2021 CS 5451 Assignm

Last Updated: 2021-09-02 Thu 10:33

CS 5451 Assignment 1:
Cor

- **Due: Fri 9/18/2021 by 11:59 pm**
- *Approximately 10% of total grade*
- Submit to Gradescope in PDF format
- You may work in groups of 2 and submit one assig

CHANGELOG: Empty

Table of Contents

- (20pts) Problem 1: Store-and-forward Routing (Gr
- (10pts) Problem 2: Shared vs. Distributed Memory
- (10pts) Problem 3: Task-dependence Graphs (Gran
- (20pts) Problem 4: Parallel Bucket Sort (Grama 3.
- (20pts) Problem 5: Parallel One-to-All Broadcast i
- (20pts) Problem 6: Bring the Heat
- Submission Instructions

(20pts) Problem 1: Store-and-forwar

Consider the routing of messages in a parallel computer t
single message of size m from P_{source} to $P_{destination}$ v
message of size m is as follows. The user breaks the mess

Test Assignment ● UNGRADED

GROUP
Turanga Leela
Phillip J Fry
View or edit group

TOTAL POINTS
- / 50 pts

QUESTION 1	
Problem 1	20 pts
QUESTION 2	
Problem 2	10 pts
QUESTION 3	
Problem 3	20 pts

7.3 Late Submission

As described in the Course Syllabus, late submissions are penalized as follows.

- On-time submissions receive no penalties
- Submitting 1-24 hours will result in the loss of 10% absolute credit
- Submitting 25-48 hours late will result in the loss 20% absolute credit
- **No submissions will be accepted more than 48 hours after a deadline.**

Author: Chris Kauffman (kauffman@cs.gmu.edu)
Date: 2021-09-08 Wed 13:54