Assignment 2 – Team Project: Distributed System and Application

Dr. Rajkumar Buyya

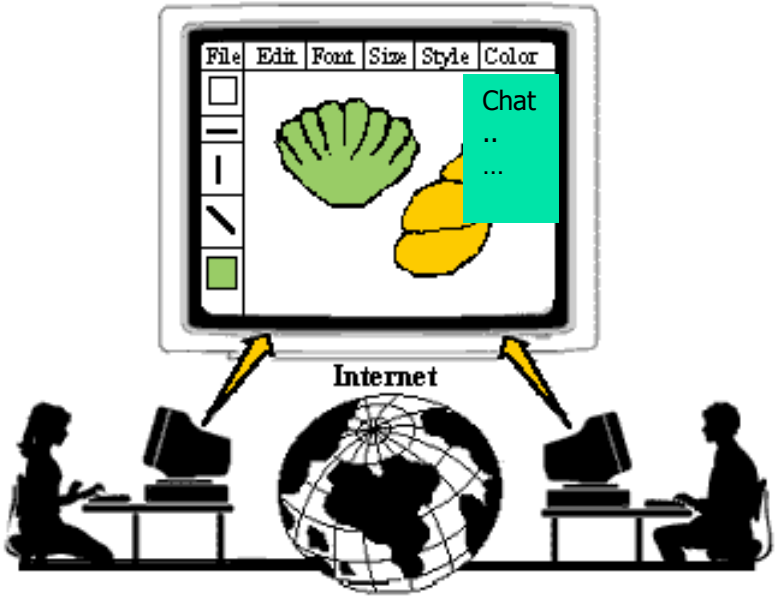
Cloud Computing and **D**istributed **S**ystems (CLOUDS) Laboratory
School of Computing and Information Systems
The University of Melbourne, Australia

Other contributors: Maria Sossa and Shashi Ilager

Distributed Shared White Board

- In these slides, we are offering mainly **guidelines** for satisfactory work, but **be innovative and creative**, which will be valued a lot.
- Team Size: 4 (Strongly recommended)
  - Note: Each member contribution will be evaluated and marked accordingly.
- General help: Ask tutors during/after tutorials in person. Also use “Discussion Board” in LMS.
- Marks Allocated: 25

Shared White Board



- Shared whiteboards allow multiple users to draw simultaneously on a canvas. There are multiple examples found on the Internet that support a range of features such as freehand drawing with the mouse, drawing lines and shapes such as circles and squares that can be moved and resized, and inserting text. In addition to these features, your implementation should include a simple chat window, that allows all the current users of the system to broadcast messages to each other.

Distributed White Board

- Develop a white board that can be shared between multiple users over the network.
- The system must be implemented in Java but you can choose the technology (e.g., it can be even Sockets) you want to use to build your distributed application:
  - Sockets
    - TCP or UDP?
    - Message format? (JSON suggested)
    - Exchange protocol?
      - E.g., client broadcasts a message with updates to all other clients, other clients reply acknowledging the message.
  - Java RMI
    - Remote Objects?
    - Remote Interface?
  - Databases for Storage
  - Please Choose any technology of your choice
    - All team members need to make sure that can achieve the goal.

Main Challenges

- Dealing with concurrency
  - Regardless of the technology you use, you will have to ensure that access to shared resources is properly handled and that simultaneous actions lead to a reasonable state.
- Structuring your application and handling the system state
  - For example you can have multiple servers that communicate with each other or a single central one that manages all the system state.
- Dealing with networked communication
  - You have to decide when/what messages are sent across the network.
  - You may have to design an exchange protocol that establishes which messages are sent in which situation and the replies that they should generate.
  - If you use RMI, then you have to design your remote interface(s) and servants
- Implementing the GUI.
  - The functionality can resemble tools like MS Paint.
  - You can use any tool/API/library you want.
  - e.g.: Java2D drawing package (http://docs.oracle.com/javase/tutorial/2d/index.html)

Requirements

- Whiteboard
  - Multiple users can draw on a shared interactive canvas.
  - Your system will support a single whiteboard that is shared between all of the clients.
  - GUI Elements:
    - Shapes: at least your white board should support for line, circle, rectangle and oval.
    - Free draw and erase must be implemented (it will be more convenient if there are several sizes of eraser)
    - Text inputting must be implemented – allow user to type text everywhere inside the white board
    - User can choose their favorite color to draw the above features. At least 16 colors should be available
    - The new drawing should be always on the top (override the old ones)
    - Chat Window (text based).
    - A “File” menu with new, open, save, saveAs and close should be provided (only the manager can control this)

Requirements

- Clients
  - Users must provide a username when joining the whiteboard. There should be a way of uniquely identifying users, either by enforcing unique usernames or automatically generating a unique identifier and associating it with each username.
  - All the users should see the same image of the whiteboard and should have the privilege of doing all the drawing operations.
  - When displaying a whiteboard, the client user interface should show the usernames of other users who are currently editing the same whiteboard.
  - Clients may connect and disconnect at any time. When a new client joins the system the client should obtain the current state of the whiteboard so that the same objects are always displayed to every active client.
  - Only the manager of the whiteboard should be allowed to create a new whiteboard, open a previously saved one, save the current one, and close the application.

Requirements

- Users should be able to work on a drawing together in real time, without appreciable delays between making and observing edits.
- There is no need to authenticate users that want to access the system.
- Important: you are NOT allowed to use ANY code taken from an existing shared whiteboard implementation.

Proposed Operational Model

- The first user creates a whiteboard and becomes the whiteboard's manager
  - java CreateWhiteBoard <serverIPAddress> <serverPort> username
- Other users can ask to join the whiteboard application any time by inputting server's IP address and port number
  - java JoinWhiteBoard <serverIPAddress> <serverPort> username
- A notification will be delivered to the manager if any peer wants to join. The peer can join in only after the manager approves
  - A dialog showing "someone wants to share your whiteboard".
- An online peer list should be maintained and displayed
- All the peers will see the identical image of the whiteboard, as well as have the privilege of doing all the operations (draw and erase)
- Online peers can choose to leave whenever they want. The manager can kick someone out at any time.
- When the manager quits, the application will be terminated. All the peers will get a message notifying them.

Guidelines

- These phases are suggestions for timely progression, you are most welcome to follow your own approach.
- Phase 1 (whiteboard) – (Milestone 1 progress demo)
  - Single-user standalone whiteboard (OR you are most welcome to implement a single user and single server).
  - Requirement A: Implement a client that allows a single user to draw all the expected elements (line, circle, rectangle, oval, freehand drawing, and erasing).
  - Requirement B: Implement the open, new, save, save as, and close functionality for a single client.

Guidelines

- Phase 2 (user management skeleton)
  - Allow the manager to create a whiteboard
  - Allow other peers to connect and join in by getting approval from the manager
  - Allow the manager to choose whether a peer can join in
  - In this phase, join in means the peer name will appear in the user list
  - Allow the joined peer to choose quit
  - Allow the manager to kick out certain peer
  - Allow the manager to close the application, and all peers get notified

Guidelines

- Phases 3 (Final – Milestone 2)
  - Integrate the whiteboard with the user management skeleton (phases 1 and 2)
  - Design issues:
    - What communication mechanism will be used?
      - Socket, RMI, or Web service, whatever you like
    - How to propagate the modification from one peer to other peers?
      - May need an event-based mechanism
    - How many threads do we need per peer?
      - At least one for drawing, one for messaging

Deliverables and Deadlines

- There are **two deadlines** for this assignment:
  - Deadline 1 – Milestone 1 (Progress Review): **Week 10 (Oct. 7-11) during you own tutorial.**
  - Deadline 2 – Milestone 2 (Final Submission): **Week 12, Friday (Oct 25) at 5:00pm.**
- **This approach ensures that Team members are working together and progressing right from the beginning.**

Milestone 1: Progress Review

- You will have a quick discussion with your tutor and explain the design of your system.
- You will show your tutor a demo of the functionality you have already implemented:
  - You should have implemented at least a single-user whiteboard (phase 1) by this time and have a clear design of your system in place.
  - Marks will be given based only on the functionality of phase 1 but to maximize your chances of completing on time, you are strongly encouraged to have made more progress by this stage (e.g. phase 1 + phase 2)
- **4 marks** will be given for demonstrating your progress in phase 1. They will be given individually, so all team members should attend.
  - Requirement A of Phase 1: 2 marks
  - Requirement B of Phase 1: 2 mark

Milestone 2: Final Submission

- Report
  - You should write a report that includes the system architecture, communication protocols and message formats, design diagrams (class and interaction), implementation details, new innovations, and a section that outlines the contribution of each member.

Name & Std. No.	Contribution area	Overall contribution (% out of 100) to Project
..	Describe…	20%? (decide reasonably)
…		15%?

  - Don't document anything you haven't implemented in the report. This is misconduct and will result in severe penalties.
- You need to submit the following via LMS:
  - Your report in PDF format *only*.
  - The *executable jar* files used to run your system's clients/server(s)
  - Your source files in a .ZIP or .TAR archive *only*.

Milestone 2: Final Submission

- Demonstrations
  - You will showcase your system and discuss your design choices during the demos (as done for assignment 1).
  - Dates and venues will be announced closer to the submission date.
  - You will be required to bring your own laptops and a printed copy of the report.

Penalties for late submissions of assignments

- Assignments submitted late will be penalized in the following way:
  - 1 day late: -1 mark
  - 2 days late: -3 marks (-1 - 2)
  - 3 days late: -6 marks (-3 - 3)
  - 4 days late: -10 marks (-6 - 4)
  - etc.